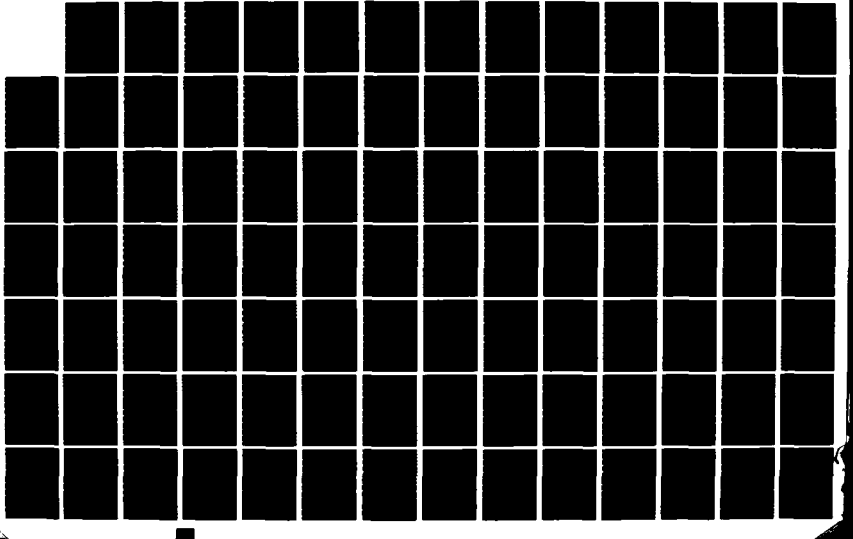
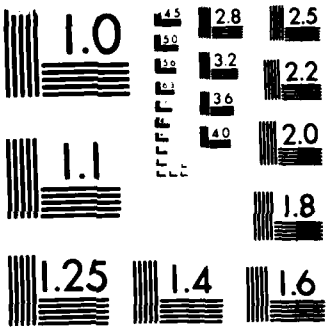
AD-A162 527 ASPECT SCAN USER'S PROGRAM FOR RCS MEASUREMENTS(U) OHIO 1/2
STATE UNIV COLUMBUS ELECTROSCIENCE LAB
A JALLOUL ET AL MAY 84 ESL-714190-7 N00014-82-K-0037
UNCLASSIFIED F/G 9/2 NL





(12)

OSU

The Ohio State University

ASPECT SCAN USER'S PROGRAM FOR RCS MEASUREMENTS

By: Amer Jalloul
Eric Walton

The Ohio State University

ElectroScience Laboratory

Department of Electrical Engineering
Columbus, Ohio 43212

Technical Report 714190-7

Contract N00014-82-K-0037

May 1984

Department of the Navy
Office of Naval Research
800 Quincy Street
Arlington, VA 22217

NOTICES

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO.	2. AD-A162 527	3. Recipient's Accession No.

4. Title and Subtitle

ASPECT SCAN USER'S PROGRAM FOR RCS MEASUREMENTS

5. Report Date
May 1984

6.

7. Author(s)
Amer Jalloul and Eric Walton

8. Performing Organization Rept. No.
ESL 714190-7

9. Performing Organization Name and Address

The Ohio State University ElectroScience Laboratory
Department of Electrical Engineering
Columbus, Ohio 43212

10. Project/Task/Work Unit No.

11. Contract(C) or Grant(G) No
(C) N00014-82-K-0037
(G)

12. Sponsoring Organization Name and Address

Department of the Navy, Office of Naval Research
800 North Quincy Street
Arlington, Virginia 22217

13. Type of Report & Period Covered
Technical

14.

15. Supplementary Notes

16. Abstract (Limit: 200 words)

This report describes a PDP-11/23 user interactive Fortran program to be used on line at the Compact Radar Range facility of the ElectroScience Laboratory at the Ohio State University. The program provides analysis of aspect angle data files with a maximum capacity of 3600 data points per file. The program is designed to interact with the user through a set of letter commands, which provide the user with a wide variety of options such as reading, writing, calibrating, subtracting and plotting different types of data files. The report includes a user's guide, a programmer's guide and listings of all the Fortran subroutines.

17. Document Analysis a. Descriptors

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement		19. Security Class (This Report)	21. No. of Pages
	This document has been approved for public release; its distribution is unlimited.	Unclassified	92
		20. Security Class (This Page) Unclassified	22. Price

(See ANSI-Z39.18) See Instructions on Reverse OPTIONAL FORM 272 (4-77) (Formerly NTIS-35) Department of Commerce

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The ElectroScience Laboratory at the Ohio State University has constructed a compact radar range facility that is capable of measuring the complex backscattered field for a variety of targets as a function of frequency and look angle [1]. The compact range system measures the backscattered signal as the target is rotated in azimuth angle by a computer controlled low cross-section pedestal support (see Figure 1 and Figure 2).

One of the problems that is encountered when the backscattered fields are measured, is the presence of undesired signal components. Such signal components include wall and ceiling reflections and some leakage of transmitter power into the receiver.

It is essential to reduce such clutter. One way of reducing this problem is by background subtraction and calibration of the data. This involves subtraction of background measurements from target measurements and then normalizing the result with respect to a sphere (see Appendix C for details on the calibration method.)

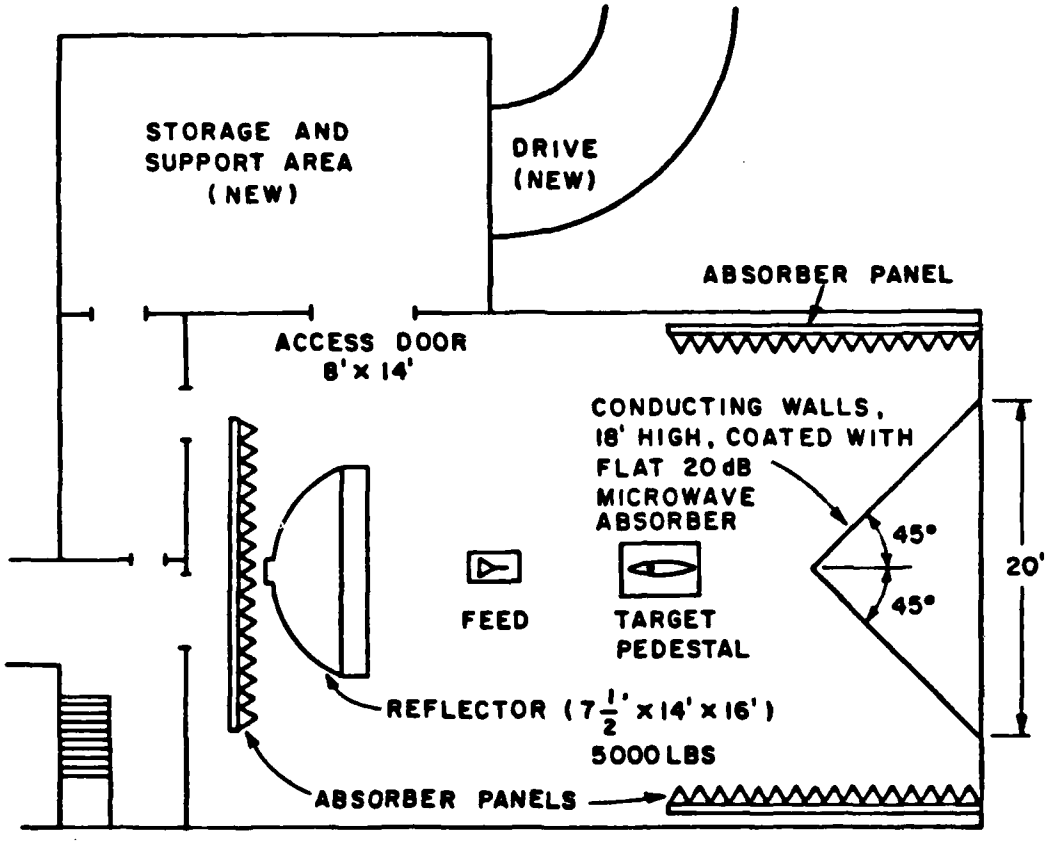


Figure 1. Anechoic room arrangement [1].

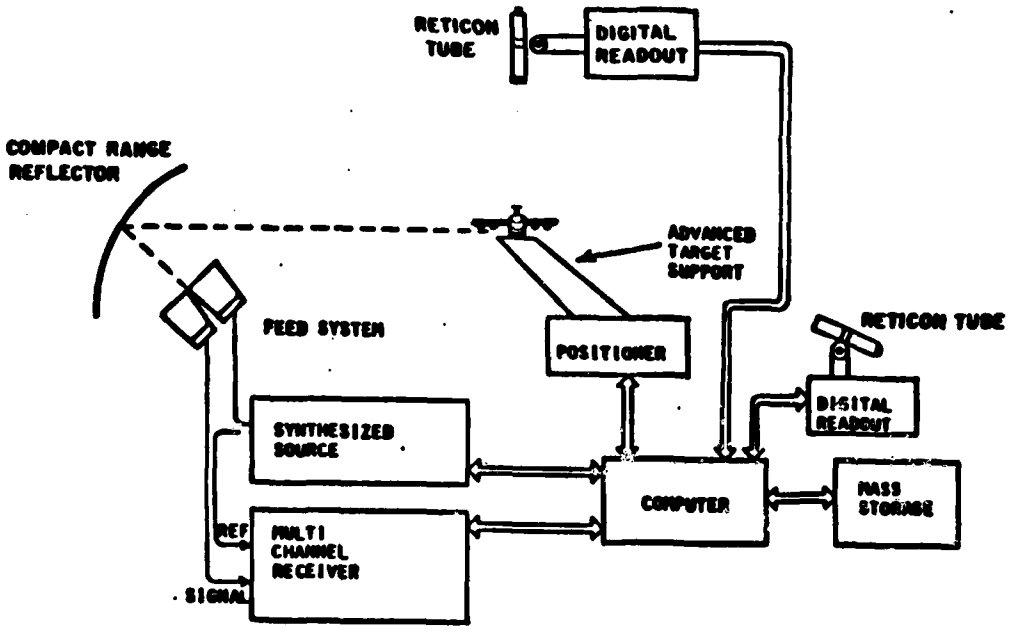


Figure 2. Schematic diagram of the compact range system [3].

The program is called ASUPRM, which stands for Aspect Scan User Program for Radar Measurements. It is a software package developed for use on the PDP-11/23. This program interacts with the user via a set of three letter command words. The program is capable of calibrating data files containing as many as 3600 data points. ASUPRM contains other options such as reading, writing, plotting and subtracting various types of data files. Table 1 is a list of the commands that are available to the user with a brief description of each.

TABLE 1

LIST OF AVAILABLE COMMANDS

1) CLB: Calibration
2) CLR: Clears CRT
3) EDH: Edit header lines
4) EXT: Exit from ASUPRM
5) LST: List of commands
6) PLT: Plot (on VT-125)
7) PRN: Print data (on CRT)
8) RDF: Read data file
9) REX: Read exact file
10) STD: Set Data
11) STF: Set Flags
12) STS: Set plotting scale
13) SUB: Subtraction
14) WDF: Write data (on default disk)

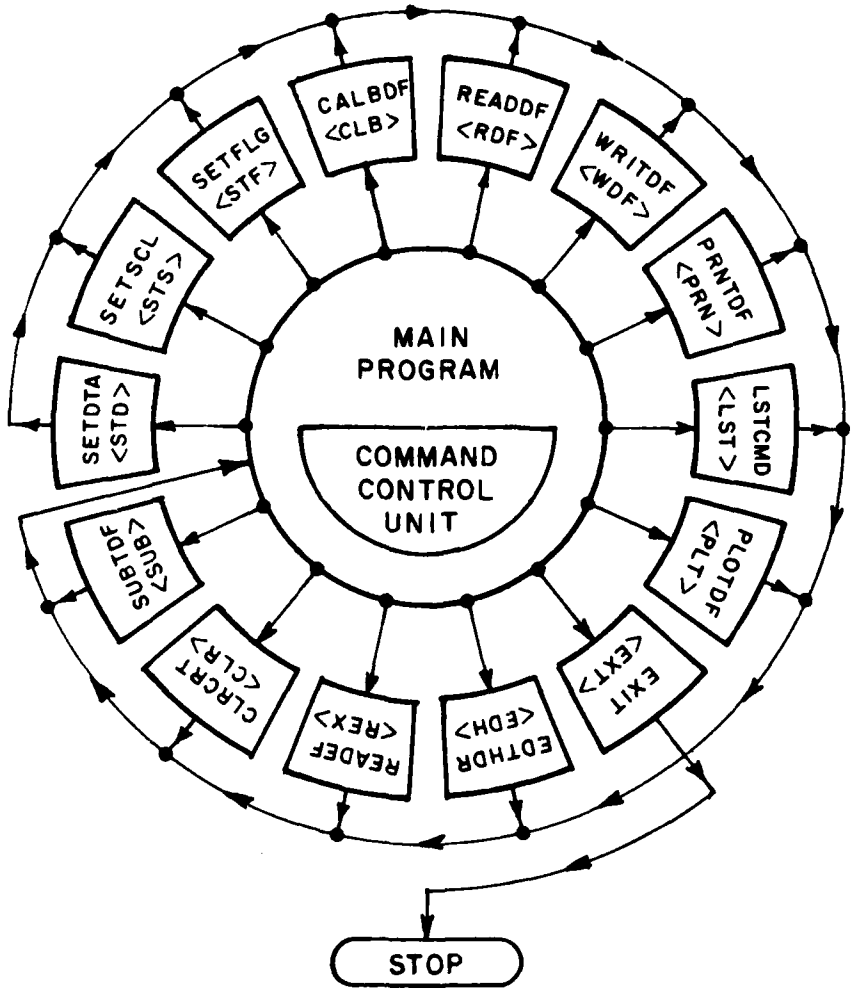


Figure 3. Flow chart of program architecture.

CHAPTER II

ASUPRM USER'S GUIDE

A. HOW TO RUN THE PROGRAM

Listed below is a step by step procedure for running ASUPRM on the PDP-11/23 machine.

1) Boot the sytstem
2) Place ASUPRM disk in DY0 *
3) Place data disk in DY1 *
4) Assign DY1 as the default disk. This can be done by typing: ASSIGN DY1 DK
5) Now, run the program. Type: RUN DY0: ASUPRM

*Note: ASUPRM disk can be placed in DY1, and the data disk in DY0 instead. Then step (4) would assign DY0 as the default disk.

When step (5) is completed, the user will be prompted with the following statement on the CRT screen,

ENTER YOUR COMMAND:

Now, any of the 14 commands can be executed by typing the proper three letter code followed by <CR>. If the user types an invalid command, the list of commands with a brief description of each will be typed on the screen, and again the statement

ENTER YOUR COMMAND:

B. THE COMMANDS

A detailed description for executing the commands is given below.

CLB: This command performs the calibration procedure on target data with the corresponding sphere and background files.

The calibration equation is:

$$\tilde{V}_{CT} = A \frac{\tilde{V}_T - \tilde{V}_B}{\tilde{V}_S - \tilde{V}_B}$$

where $\tilde{V}_{CT}$, $\tilde{V}_T$, $\tilde{V}_B$ and $\tilde{V}_S$ are phasor quantities.

$\tilde{V}_{CT}$: is the calibrated data

$\tilde{V}_T$: is the target data

$\tilde{V}_B$: is the background data

$\tilde{V}_S$: is the sphere (or other calibration target) data

A: is the exact value of the RCS of the sphere (or other calibration target) in (dBsm or dBscm), it is a scalar.

There are two ways to execute the calibration subroutine. One way is to simply type, as a command, "CLB". The program then will ask the user to enter the names of the three files involved in the calibration and the value of the exact file in the following order:

< > ENTER SPHERE FILE NAME:

< > ENTER BACKGROUND FILE NAME:

< > ENTER TARGET FILE NAME:

< > ENTER THE VALUE OF THE EXACT FILE:

When entering the sphere file name, the calibration subroutine delays reading the file whereas target and background files are read in (by calling subroutine READDF) when their file names are entered.

The other way to perform calibration is to first execute the command "STD" (see details on "STD"). After "STD" is executed, type command "CLB", which automatically extracts the information given when "STD" was executed and then computes the calibrated data. When calibration is done, the user will be asked whether he or she desires to modify the header information and will be also asked whether he or she wants to write (on the default disk) the calibrated data.

The reason for developing two procedures for executing the calibration subroutine is for the user's convenience. Occasionally, it is desired to calibrate many different target files with respect to the same sphere, background and exact files, hence it becomes redundant to enter the same information every time calibration is performed.

CLR: This command clears the CRT screen. It is often used after plotting a data file.

EDH: Use of this command permits the user to modify an existing header. When "EDH" is executed, all three lines are printed on the screen and the user is asked which line to is be edited. Consider the following example.

```
ENTER YOUR COMMAND:  EDH

Line 1:  I AM A DATA FILE CALLED TARGET
Line 2:  THIS IS LINE NUMBER TWO
Line 3:  THIS IS LINE NUMBER THREE

Line ?  1

Line 1:  I AM A DATA FILE CALLED TARGET
         ^NOT#          %%%%%%%%%%%%%
```

Line 1: I AM NOT A DATA FILE

Line ? <CR>

ENTER YOUR COMMAND:

Use <CR> to exit from this subroutine at request of line number. Use "%" to delete characters and "^" to insert with "#" at the end of inserted word.

EXT: This command exits from ASUPRM gracefully.

LST: This command supplies the user with a list of all the available commands with a brief description of each. There are 14 commands. The command list will also be printed on the CRT any time an illegal command is used by the user.

PLT: This command allows the user to plot data from the virtual memory array data (files can be target, background or sphere files.) When executing this command, the user is asked to enter the type of file (a set of two letter codes, TF, BF, SF for target background and sphere files respectively).

When the file is specified, the computer types the name of the file and asks the user if he/she desires to plot that file or whether to plot another. When the desired file name is agreed and minimum values of amplitude and phase are computed and typed out on the screen as follows

MAXIMUM AMPLITUDE =

MINIMUM AMPLITUDE =

MAXIMUM PHASE =

MINIMUM PHASE =

The computer will then ask the user the following:

DO YOU DESIRE TO SET YOUR OWN SCALE?
IF YES, TYPE "Y". IF NOT PUSH RETURN.

If the user decides to set a particular scale, then the subroutine which sets the scale will be called. If, however the user decides not tc choose a particular scale, then an automatic scale adjustment will occur. This scale causes the maximum value to be rounded up to the next factor of ten, and the minimum value to be rounded down to the previous factor of ten. An example is illustrated in Figure 4.

If the scale was previously set by executing "STS", then it will display the scale values, type out the maximum and minimum values of the data to be plotted and then ask the user whether he/she still desires the existing scales. If the user decides to change them, he/she may do so, and the subroutines which set the scale will be called again.

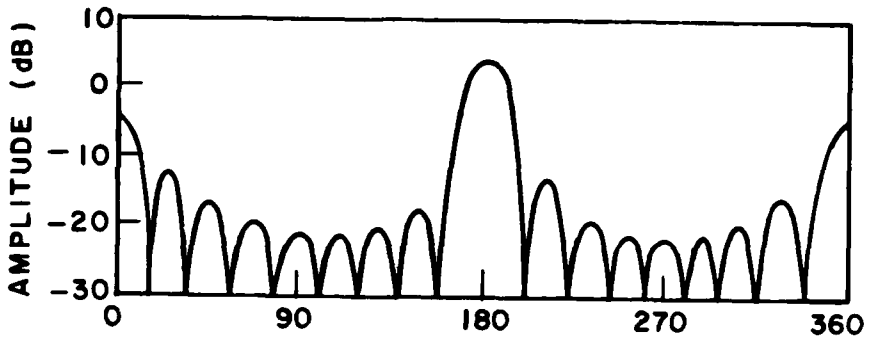


Figure 4. An example of an amplitude plot for a data file. Maximum amplitude is +3dB and minimum amplitude is -30dB.

The user will be required to specify the range of aspect angle for plotting, enabling the user to stretch the horizontal scale as desired. The plot will consist of two graphs, one for amplitude and the other for phase, both as a function of aspect angle. During the plotting procedure, the subroutine can be interrupted by pushing the carriage return (<CR>). If the user does so, the plotting procedure halts. If the letter "Q" is typed, the plotting subroutine is terminated and the user goes back to command mode. (At this stage the screen may be cleared by typing "CLR".) If, however instead of typing "Q", the user types any other letter or <CR>, the plotting resumes from where it was interrupted.

PRN: This command permits the user to get a printout on the CRT screen of a specified portion of a data file. To be able to execute "PRN", you must have the desired data in virtual memory. This may be accomplished by reading in the data file using "RDF" command.

A range of aspect angles must be specified:

$$\alpha_1 < \alpha < \alpha_2$$

When "PRN" is executed, the computer asks the user for the value of α, and α_2 in the following order:

ENTER VALUE OF ALPHA1:

when entered

ENTER VALUE OF ALPHA2:

When entered, a list of aspect angles and the corresponding data point - amplitude and phase - is printed on the screen.

RDF: This command allows a data file to be read into the virtual memory. A data file can be a target, background or a sphere file, depending on the specification.

Upon executing "RDF", the user will be prompted with the following:

```
TARGET           : TF
BACKGROUND FILE  : BF
SPHERE FILE      : SF

ENTER TYPE OF FILE:
```

The user should use the above two letter codes to specify the data file. When the file is specified, the following statement appears:

```
                TARGET
                  or
< > ENTER       BACKGROUND     FILE NAME:
                  or
                SPHERE
```

REX: This command allows the exact calibration value to be read in. This value is a single number, usually in "dBsm" (decibels above a square meter). This number must be supplied by the user in units of "dBsm". (Decibels above a square centimeter "dBscm" is also acceptable.)

STD: When this command is executed, data file names of the target, sphere and background with the value of the exact file are stored in a buffer. When "CLB" is entered as a command by the user, the calibration subroutine automatically extracts the required information from the buffer. (Makes calibration more enjoyable.)

Here is how it works.

When STD is typed, the computer types the following:

```
TARGET          (1)

BACKGROUND      (2)

SPHERE          (3)

EXACT VALUE     (4)

LISTING         (5)


OPTION          (?)
```

The computer now is waiting for an option to be selected between (1) and (5). If 0 or <CR> is typed, the system goes back to command mode. If the option number is more than 5, the program asks for the option again.

As an example, say the user selects option 1, the computer will request the following:

```
ENTER TARGET FILE NAME:
```

Similarly, for options 2, 3 and 4. If option 5 is selected, the list of file names is typed on the screen, i.e.:

TARGET : A3242C

BACKGROUND: A3242A

SPHERE : A3242B

EXACT FILE: -17.4

STF: This command permits the user to reset the current status of the flags.

What are the flags?

ASUPRM uses six integer variables to check for specific information. Basically the information indicates whether a particular subroutine was executed or not. If a subroutine was executed, the value of the flag is 1, otherwise its value is 0. The flags are typed as a six digit integer number.

Flags = I_1 I_2 I_3 I_4 I_5 I_6

Each value I_k, k = 1, 2, . . ., 6, determines the state of subroutine Sk. If I_k = 1 then S_k was executed if I_k = 0, then S_k was not executed.

(a) Flag #(1), i.e. I, determines whether a target file has been defined.

(b) Flag #(2), i.e. I_2, determines whether a background file has been defined.

(c) Flag #(3), i.e. I_3, determines whether a sphere file has been defined.

(d) Flag #(4), i.e. I_4, determines whether a value for the exact file has been defined.

(e) Flag #(5), i.e., I_5, determines whether a plotting scale has been defined.

(f) Flag #(6), i.e. I_6, determines whether file names exist in the buffer used by the calibration subroutine.

STS: This command allows the user to set the plotting scale. When "STS" is used the following values are defined:

MAXIMUM AMPLITUDE =

MINIMUM AMPLITUDE =

MAXIMUM PHASE ANGLE =

MINIMUM PHASE ANGLE =

In case the user makes a typing error the routine asks if any typing errors were made so that the user can enter the values again.

Once the scale is set, it is valid for subsequent plots, unless specified in the plotting subroutine.

SUB: This command permits the subtraction of two data files. When "SUB" is entered, the following statement is typed:

< File "Y" > = < File "Y" > - < File "X" >

< > ENTER NAME OF FILE "X":

When "X" file name is given, the computer asks

< > ENTER NAME OF FILE "Y":

When subtraction is completed, the user is asked if he/she desires to change the header, and then whether to write the file (on the default disk).

WDF: This command allows a specified data file to be written on the default disk. The data must already exist in the virtual memory before executing "WDF".

The user must speficy the type of file to be written according to the following two letter codes: TF, BF, SF, and CF which stands for target, background, sphere and calibrated file.

CHAPTER III

PROGRAMMER'S GUIDE

Circumstances might arise where a particular user needs to expand the total number of data points, i.e., decrease the increment in aspect angle. From appendix (A), the total number of virtual memory is 65,536 bytes.

Thus, the maximum number of data points that can be stored in one data file is

$$N_p = \frac{65,536 - 360}{8} = 8,147 \text{ points}$$

each point consisting of an amplitude and phase.
Therefore, the minimum increment in aspect angle is $\Delta\theta_{min}$.

$$\Delta\theta_{min} = \frac{360}{8,147-1} \cong 0.0442 \text{ (degrees)}$$

If we choose $\Delta\theta = 0.05°$, this corresponds to an array of:

$$2.\left(\frac{360}{0.05 +1}\right) = 7201 \times 2 \text{ elements}$$

Therefore, the virtual arrays that were used UT(3604,2) and UB (3604,2) will have to be replaced by another virtual array; ARRAY (7201,2).

Doing so, the algorithm for the calibration subroutine should be modified. Figure 5 shows the modification necessary in the algorithm.

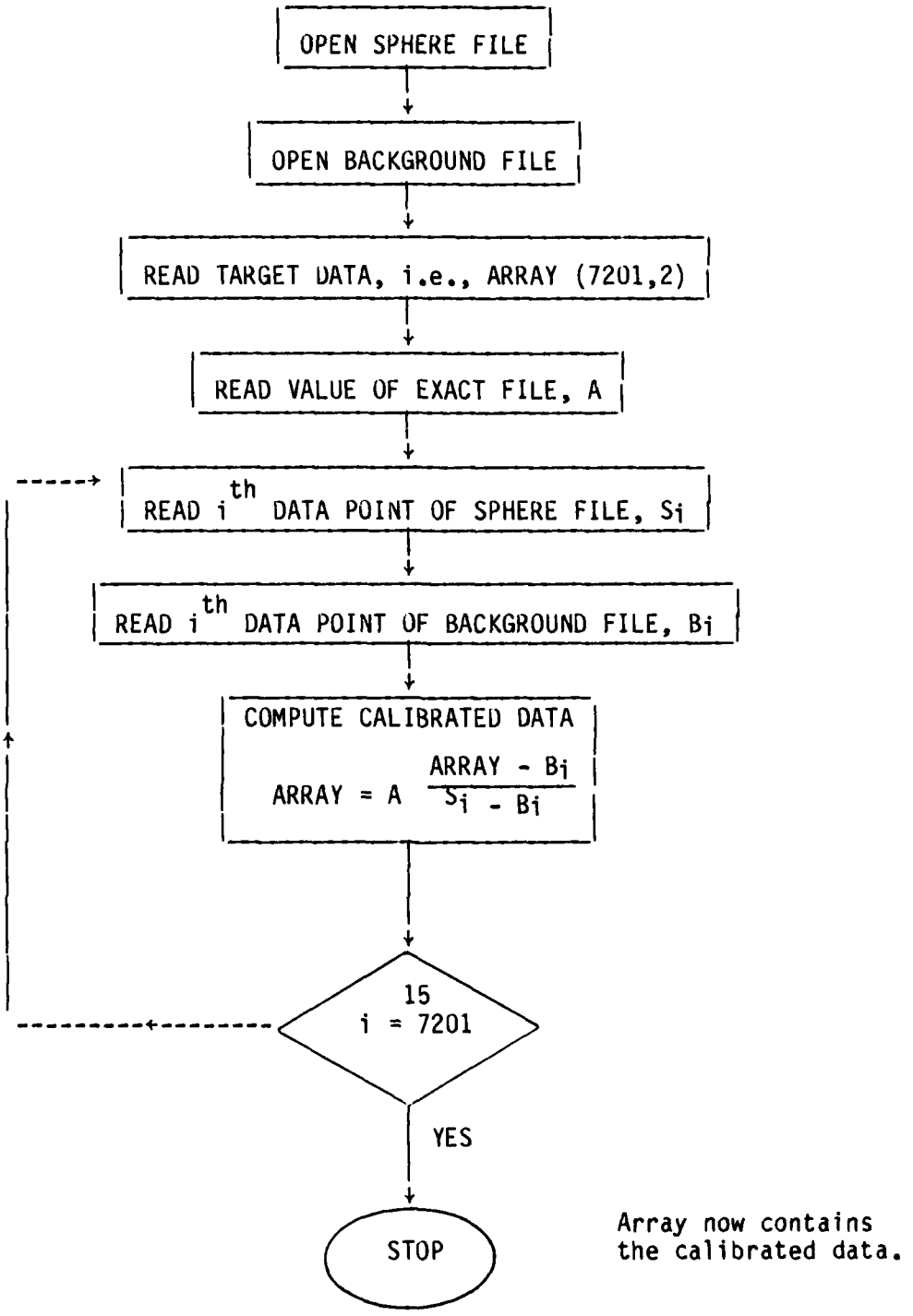


Figure 5. A modified algorithm for calibration.

CONCLUSIONS

In summary, ASPURM gives a wide variety of options for the manipulation of aspect angle data files. Such options include calibration, subtaction, plotting, reading and writing various types of data files. The program was also designed to permit modification and/or addition of new command options (see discussion in programmer's guide). An effort was made to prevent program failure during operation (e.g., upon the use of illegal commands or parameters by the user).

APPENDIX A

DATA FORMAT

The standard form of a data file stored on a floppy disk is given in Tables A-1 and A-2 [2].

TABLE A-1

DATA ARRANGEMENT OF A STANDARD FILE [2]

FILE HEADER	LINE 1 (60 characters) LINE 2 (60 characters) LINE 3 (60 characters
DATA POINT 1	Amplitude in dB, phase in degrees
DATA POINT 2	Amplitude in dB, phase in degrees
	.
	.
	.
	.
	.
	.
DATA POINT 801	

TABLE A-2

DETAILED BLOCK AND BYTE ASSIGNMENT FOR THE DATA FILES [2]

NOTE: Line 1 ≡ 1st character ~ 60th character
Line 2 ≡ 61st character ~ 120th character
Line 3 ≡ 121st character ~ 180th character

BLOCK 1 BYTE #	(8 BIT BYTES)
1	1st Character
2	NOT USED
3	2nd Character
4	NOT USED
	.
	.
	.
	.
2n-1	nth Character
	.
	.
	.
	.
359	180th Character
360	NOT USED
361	Amplitude of 1st
362	Data Point
363	(Four-Byte Real
364	Number)
365	Phase of 1st
366	data point
367	(Four-Byte Real
368	Number)
	.
	.
	.

APPENDIX B

STORAGE CAPACITY REQUIREMENT

Why 3600 data points?

The PDP - 11/23 virtual memory capacity is 64k bytes, call it $C_{v,max}$.

$$C_{v,max} = 64\ (2^{10}) = 65{,}536 \text{ bytes}$$

Let the total number of data points (a data point contains both amplitude and phase) possible to manipulate be M_v.

In a standard data file, each data point takes up eight bytes - four bytes for amplitude and four bytes for phase. A data file also contains header information, they take up 360 bytes of memory.

If the total number of bytes/data file is N_T:

$$N_T = 8\ M_v + 360$$

The calibration algorithm that ASUPRM uses requires reading in two data files and opening the header for one.

$$\Rightarrow C_{v,max} = 2(8M_v + 360) + 360.$$

But

$$C_{v,max} = 65{,}536 \text{ bytes}$$

$$\Rightarrow 65{,}536 = 16M_V + 1080$$

$$\Rightarrow M_V = \frac{65{,}536 - 1080}{16}$$

$$M_V \approx 4028$$

Let the increment in aspect angle be $\Delta\theta_A$ in degrees.

$$\Rightarrow \frac{360}{\Delta\theta_A} + 1 \leq M_V$$

has to be satisfied.

$$\Rightarrow \Delta\theta_A \geq \frac{360}{M_V - 1}$$

and using the result that $M_V \approx 4028$,

$$\Delta\theta_A \geq \frac{360}{4028-1} \approx 0.0894$$

$$\Rightarrow \Delta\theta_A \geq 0.0894$$

Thus, if we choose a nice number for $\Delta\theta_A$, $\Delta\theta_A = 0.1$

$\Rightarrow$ Number of data points becomes

$$\frac{360}{0.1} + 1 = \underline{3601}$$

and $\Delta\theta_A \geq 0.0894$ is still satisfied.

APPENDIX C

CALIBRATION EQUATIONS

Terminology =

A (dBsm) = Exact RCS value of sphere (or other calibration target)

$\tilde{V}_T = V_T e^{j\theta T}$ = Target data

$\tilde{V}_B = V_B e^{j\theta B}$ = Background data

$\tilde{V}_S = V_S e^{j\theta S}$ = Sphere data (or other calibration target data)

$\tilde{V}_C = V_C e^{j\theta C}$ = Calibrated data

The calibrated vector is given by:

$$\tilde{V}_C = V_C e^{j\theta C} = A \frac{\tilde{V}_T - \tilde{V}_B}{\tilde{V}_S - \tilde{V}_B} \qquad \text{(C-1)}$$

This results in:

$$V_C = A \left[\frac{a_1^2 + b_1^2}{a_2^2 + b_2^2} \right]^{1/2} \qquad \text{(C-2)}$$

$$\theta_C = \arctan\left(\frac{b_1}{a_1}\right) - \arctan\left(\frac{b_2}{a_2}\right) \qquad \text{(C-3)}$$

where

$$a_1 = V_T \cos \theta_T - V_B \cos \theta_B$$

$$b_1 = V_T \sin \theta_T - V_B \sin \theta_B \qquad \text{(C-4)}$$

$$a_2 = V_S \cos \theta_S - V_B \cos \theta_B$$

$$b_2 = V_S \sin \theta_S - V_B \sin \theta_B$$

(*) Data are stored on disks with the amplitude in units of (dB's) and the phase in degrees, so the above equations assume that amplitudes are converted from dB's to volts.

(**) In equation (C-3), the arc tangent function is a four-quadrant arc tangent function.

APPENDIX D
LINKER SEQUENCE

LINKER COMMAND SEQUENCE LIST

ASUPRM

WRITDF

SUBWRI

EDTHDF

SUBHDR

CALBDF

SUBTDF

PRNTDF

PNTOUT

SETFLG

SETDTA

SETSCL

MINMAX

LSTCMD

READDF

SUBREA

EXIT

READNM

PLOTDF

NUMBER

FRAME

CLRCRT

PLACEP

DRLIB

APPENDIX E

ORIGINAL SUBROUTINES FOR ASUPRM

The list below, is of the Fortran programs (subroutines) that were developed specifically for ASUPRM (not including the subroutines written before ASUPRM was developed). These subroutines are:

1) READDF
2) SUBREA ($ARRAY_{,k}$)
3) WRITDF
4) SUBWRI ($ARRAY_{,k}$)
5) READNM
6) PRNTDF
7) PNTOUT
8) LSTCMD
9) EXIT
10) CLEAR
11) EDTHDF
12) PLOTDF
13) MINMAX
14) SETSCL
15) SETDTA
16) SETFLG
17) CALBDF
18) SUBTDF
19) ASUPRM

* Note that SUBREA (ARRAY$_{,k}$) and SUBWRI(ARRAY$_{,k}$) are modifications of previously written subroutines called SUBREA (ARRAY) and SUBWRI (ARRAY) respectively. The subscript k, takes on values of 1, 2, 3, 4 and 5 for target, background, sphere, Y or X files respectively.

APPENDIX F

ASUPRM PROGRAM

```
C*************************************************************************
C*********************   THIS IS THE MAIN PROGRAM   **********************

C*************************************************************************
C
      VIRTUAL UT(3604,2),UB(3604,2)
C
      CALL CLRCRT
      TYPE 77
77    FORMAT(,/
     +  ,10X,'  *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*',/
     +  ,10X,'<<  ASPECT SCAN USER PROGRAM FOR RADAR MEASUREMENTS  >>',/
     +  ,10X,'  *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*',///
     +  ,10X,'     ***     ****   *   *   ****   ****   **   **',/
     +  ,10X,'    *   *   *       *   *   *   *  *   *  * * * *',/
     +  ,10X,'    *****    ***    *   *   ****   ****   *  *  *',/
     +  ,10X,'    *   *       *   *   *   *      *  *   *     *',/
     +  ,10X,'    *   *   ****    *****   *      *   *  *     *',////)
21    II=0
C    THE TWO LINES BELOW CAUSE THE SYSTEM TO BEEP !
C
369   FORMAT(10A1)
      TYPE 369,' ',1799
C
20    TYPE 80
80    FORMAT(//,'  **> ENTER YOUR COMMAND **> : ',$)
      ACCEPT 73,CMD
73    FORMAT(A3)
      IF(CMD.EQ.'STF')GOTO 40
      IF(CMD.EQ.'STD')GOTO 42
      IF(CMD.EQ.'SUB')GOTO 48
      IF(CMD.EQ.'EDH')GOTO 50
      IF(CMD.EQ.'PRN')GOTO 100
      IF(CMD.EQ.'EDH')GOTO 111
      IF(CMD.EQ.'REX')GOTO 222
      IF(CMD.EQ.'RDF')GOTO 333
      IF(CMD.EQ.'WDF')GOTO 666
      IF(CMD.EQ.'CLB')GOTO 777
      IF(CMD.EQ.'PLT')GOTO 888
      IF(CMD.EQ.'LST')GOTO 999
      IF(CMD.EQ.'EXT')GOTO 1000
      IF(CMD.EQ.'CLR')GOTO 1100
      IF(CMD.EQ.'STS')GOTO 1200
      TYPE 35
35    FORMAT(/,' © ILLEGAL USE OF COMMAND,TRY AGAIN .',)
      II=II+1
      IF(II.EQ.4)GOTO 1000
      GOTO 999
40    CALL SETFLG
      GOTO 21
```

```
42      CALL SETDTA
        GOTO 21
48      CALL SUBTDF(UT,UB)
        GOTO 21
50      CALL EDTHDF
        GOTO 21
100     CALL PRNTDF(UT,UB)
        GOTO 21
111     CALL SUBHDR
        GOTO 21
222     CALL READNM
        GOTO 21
333     CALL READDF(UT,UB)
        GOTO 21
666     CALL WRITDF(UT,UB)
        GOTO 21
777     CALL CALBDF(UT,UB)
        GOTO 21
888     CALL PLOTDF(UT,UB)
        GOTO 21
1100    CALL CLRCRT
        GOTO 21
1200    CALL SETSCL
        GOTO 21
999     CALL LSTCMD
        GOTO 20
1000    CALL EXIT
        STOP
        END
```

```
C*************************************************************************

C***************   THIS SUBROUTINES LISTS THE COMMANDS  ****************

C*************************************************************************

C
      SUBROUTINE LSTCMD
      TYPE 10
10    FORMAT('     <** COMMANDS AVAILABLE FOR USER **> ',/)
      TYPE 11
11    FORMAT(' ***********************************************',)
      TYPE 12
12    FORMAT(' 1)  LST : Lists the commands available for user .',/
     +  ,' 2)  RDF : Reads a data file from Default Disk .',/
     +  ,' 3)  REX : Reads the multiplying factor .',/
     +  ,' 4)  EDH : Edits header information of a data file .',/
     +  ,' 5)  PLT : Plots a data file on the "VT-100" .',/
     +  ,' 6)  WDF : Writes a data file on the Default Disk .',/
     +  ,' 7)  PRN : Writes out a specified portion of a data',/
     +  ,'           file on the "CRT" screen .',/
     +  ,' 8)  CLB : Calibrates data file .',/
     +  ,' 9)  SUB : Subtracts a data file from another .',/

     +  ,' 10) CLR : Clears the "CRT" screen .',/
     +  ,' 11) STS : Allows the user to set his own plotting scale .',/
     +  ,' 12) STD : Allows the user to set his data files for ',/
     +  ,'           calibration at once .',/
     +  ,' 13) STF : Allows the user to control the main program ',/
     +  ,'           by setting values for the flags as desired .',/
     +  ,' 14) EXT : Exits from the program completely .',/
     +  ,'***********************************************',/,$)
      RETURN
      END
```

```
C*************************************************************************

C***************  THIS SUBROUTINE READS A DATA FILE  ********************

C*************************************************************************

C
      SUBROUTINE READDF(UT,UB)
      VIRTUAL UT(3604,2),UB(3604,2)
      INTEGER FT
      COMMON/KRSB/IJK
      COMMON/FKIND/FT
      COMMON/FLAGS/MT,MB,MS,MN,MST,MSD
C
C   IJK : IS A VARIABLE WHICH TAKES ON A VALUE OF "1" IN CASE OF ERROR
C         IN READING THE DATA FILE ,AND A VALUE OF "0" OTHERWISE .
C
      MSD=0
C
11    TYPE 12
12    FORMAT(/,'    TF  : TARGET FILE       ',/
     +  ,'    BF  : BACKGROUND FILE      ',/
     +  ,'    SF  : SPHERE FILE    ',//,$)
      TYPE 13
13    FORMAT(/,'    SPECIFY FILE TYPE :  ',$)
      ACCEPT 14,FT
14    FORMAT(A2)
      IF(FT.EQ.'TF')GOTO 45
      IF(FT.EQ.'BF')GOTO 50
      IF(FT.EQ.'SF')GOTO 51
      TYPE 77
77    FORMAT(/,' ILLEGAL FILE SPECIFICATION ,TRY AGAIN .',/,$)
      GOTO 11
45    MT=1
      CALL SUBREA(UT,1)
      IF(IJK.EQ.1)GOTO 25
      RETURN
25    IJK=0
      RETURN
50    MB=1            ! SETTING FLAGS .
      MS=0
      CALL SUBREA(UB,2)

      IF(IJK.EQ.1)GOTO 25
      RETURN
51    MS=1            ! SETTING FLAGS .
      MB=0
      CALL SUBREA(UB,3)
      IF(IJK.EQ.1)GOTO 25
      RETURN
      END
```

```
      SUBROUTINE SUBREA(ARRAY,K)
C
C     ************************************
C     THIS ROUTINE READS AN OLD DATA FILE
C     FROM THE USER DISC. STATUS WORDS FOR
C     PLOTTING AND SUBTRACTION ARE RESET.
C     CALLED FROM:SYSTEM
C     CALLS:NONE
C     ************************************
C
C     VARIABLES
C
      BYTE LINE1(60),LINE2(60),PARAM(60)
      BYTE FNMT(31),FNMB(31)
      COMMON /KRSB/IJK
      COMMON /FLG/ FLAG
      COMMON /FILE/ITS1,KT,ISTYP,IFLG
      COMMON /LISFRQ/ LOFQ,IUPFQ,INCRE
      COMMON /MHDR/ LINE1,LINE2,PARAM,HDR
      COMMON /MWRI/ FNMT,FNMB
      COMMON /SFIND/ STRG,KSTRG,KCHR
C
      COMMON/FLAGS/MT,MB,MS,MN,MST,MSD
C
      DOUBLE PRECISION STRG(25)
      INTEGER KCHR(25)
      LOGICAL HDR
      VIRTUAL ARRAY(3604,2)
C
C     FORMATS
C
002   FORMAT (31A1)
003   FORMAT (' ',60A1)
004   FORMAT (I4)
005   FORMAT (I5)
010   FORMAT (' <><> ENTER TARGET FILE NAME : ',$)
011   FORMAT (' <><> ENTER BACKGROUND FILE NAME : ',$)
012   FORMAT (' <><> ENTER SPHERE FILE NAME : ',$)
013   FORMAT (' <><> ENTER NAME OF FILE "X" : ',$)
014   FORMAT (' <><> ENTER NAME OF FILE "Y" : ',$)
050   FORMAT (' Open Error--File Does Not Exist.')
051   FORMAT (' Decode Error--Line Counter "KT".')
052   FORMAT (' --- Read Aborted ---')
88    FORMAT (I2)
C
C
      IF(K.EQ.4)GOTO 478
      IF(K.EQ.5)GOTO 479
      IF(MSD.EQ.1)GOTO 450
      GOTO 110
478   TYPE 013
```

```
        GOTO 2220
479     TYPE 014
        GOTO 1110
        IF(MSD.EQ.1)GOTO 450
        GOTO 110
450     IF(K-2)1109,2219,335
110     IF(K-2)111,222,333
111     TYPE 010
1110    ACCEPT 002,FNMT
1109    IF (FNMT(2).EQ.32) TYPE 052
        IF (FNMT(2).EQ.32) GOTO 351
        GOTO 444
351     IJK=1
        RETURN
222     TYPE 011
2220    ACCEPT 002,FNMB
2219    IF (FNMB(2).EQ.32) TYPE 052
        IF (FNMB(2).EQ.32) GOTO 351
        GOTO 555
333     TYPE 012
334     ACCEPT 002,FNMB
335     IF(FNMB(2).EQ.32) TYPE 052
        IF(FNMB(2).EQ.32) GOTO 351
        GOTO 555
C
C       OPEN FILE
C
444     OPEN (UNIT=13,NAME=FNMT,TYPE='OLD',FORM='UNFORMATTED',
     &  READONLY,ERR=1000)
        GOTO 567
555     OPEN (UNIT=13,NAME=FNMB,TYPE='OLD',FORM='UNFORMATTED',
     &  READONLY,ERR=1000)
C
567     READ (13) LINE1
        READ (13) LINE2
        READ (13) PARAM
        TYPE 003,LINE1
        TYPE 003,LINE2
        TYPE 003,PARAM

        HDR=.TRUE.
C
C       DECODE HEADER
C
        IF (PARAM(8).EQ.'F') GO TO 995
        DECODE (4,004,PARAM(4),ERR=1001) KT
        DECODE (5,005,PARAM(12),ERR=1001) LOFQ
        DECODE (5,005,PARAM(21),ERR=1001) INCRE
        DECODE (2,88,PARAM(31),ERR=1008) ISTYP
        GO TO 388
```

```
995     DECODE (4,2004,PARAM(4),ERR=1001) KT
2004    FORMAT (I3)
        DECODE (5,005,PARAM(11),ERR=1001) LOFQ
        DECODE (5,005,PARAM(20),ERR=1001) INCRE
        ISTYP=0
C
C
C       SET BAKAVE FLAGS AND TYPES [SCN TYPE]
C
388     IFLG=0
        IF (MOD(ISTYP,10).NE.3) GO TO 389
        IFLG=2
389     CNE=INCRE/100.
        IUPFQ=IFIX(LOFQ+(KT-1)*CNE)
C
        FMIN=LOFQ
        FMAX=IUPFQ
C
        DO 200 K1=1,KT
          READ (13) ARRAY(K1,1),ARRAY(K1,2)
200     CONTINUE
C
C       GET AVE VALUES IF PRESENT [JUNK OTHERWISE]
        IF (PARAM(8).EQ.'F') GO TO 1004
C
        READ (13) ARRAY(3603,1),ARRAY(3603,2)
        READ (13) ARRAY(3604,1),ARRAY(3604,2)
C
1004    CLOSE (UNIT=13,DISP='SAVE')
        RETURN
1000    TYPE 050
        IJK=1
        RETURN
1001    TYPE 051
        IJK=1
        RETURN
1008    TYPE *,'Old File Type'
        ISTYP=-1
        GO TO 388
        END
```

```
C*************************************************************************

C*********  THIS SUBROUTINE READS THE EXACT FILE VALUE  'AE' . *********

C*************************************************************************

C
        SUBROUTINE READNM
        COMMON/FLAGS/MT,MB,MS,MN,MST
        COMMON/MULTF/AE
        MN=1
21      TYPE 23
23      FORMAT(//,' ENTER EXACT FILE VALUE = ',$)
22      FORMAT(F8.2)
        READ(5,22,ERR=28)AE
        RETURN
28      TYPE 19
19      FORMAT(/'   (*) ILLEGAL FORMAT SPECIFICATION (*)')
        GOTO 21
        RETURN
        END
```

```
C*************************************************************************

C*****************  THIS SUBROUTINE WRITES A DATA FILE  ****************

C*************************************************************************

C
      SUBROUTINE WRITDF(UT,UB)
      VIRTUAL UT(3604,2),UB(3604,2)
      INTEGER FK
C
      COMMON/FLAGS/MT,MB,MS,MN,MST
      COMMON/DDI/FT
C
66    TYPE 11
11    FORMAT(//,'    TARGET FILE     : TF ',/
     +  ,'    BACKGROUND FILE : BF ',/
     +  ,'    SPHERE FILE     : SF ',/
     +  ,'    CALIBRATED FILE : CF ',//,$)
      TYPE 12
12    FORMAT(/,'    SPECIFY FILE TYPE  :  ',$)
      ACCEPT 15,FT
15    FORMAT(A2)
      IF(FT.EQ.'TF')GOTO 22
      IF(FT.EQ.'BF')GOTO 33
      IF(FT.EQ.'SF')GOTO 34
      IF(FT.EQ.'CF')GOTO 22
      TYPE 88
88    FORMAT(/,' (*) ILLEGAL FILE SPECIFICATION ,TRY AGAIN .',/,$)
      GOTO 66
22    CALL SUBWRI(UT,1)
      RETURN
33    CALL SUBWRI(UB,2)

      RETURN
34    CALL SUBWRI(UB,3)
      RETURN
      END
```

```
      SUBROUTINE SUBWRI(ARRAY,L)
C
C     ***********************************************
C     THIS WRITES A FILE
C
C     ***********************************************
C
C     VARIABLES
C
      BYTE LINE1(60),LINE2(60),PARAM(60)
      BYTE FNMT(31),FNMB(31),FNMS(31)
      COMMON /MHDR/ LINE1,LINE2,PARAM,HDR
      COMMON /MWRI/ FNMT,FNMB,FNMS
      COMMON /FILE/ITS,KT,ISTYP,IFLG
      LOGICAL HDR,ANS
      REAL ARRAY
      VIRTUAL ARRAY(3604,2)
C
      ANS=.TRUE.       !SET FOR LAST ABORT
C
C     FORMATS
C
001   FORMAT (' ')
002   FORMAT (31A1)
004   FORMAT (' ',60A1)
009   FORMAT (' <> ENTER TARGET FILE NAME : ',$)
008   FORMAT (' <> ENTER BACKGROUND FILE NAME : ',$)
007   FORMAT (' <> ENTER SPHERE FILE NAME : ',$)
050   FORMAT (' Open Error -- Data In FTN13.DAT')
051   FORMAT (' --- No Data Available ---')
052   FORMAT (' --- No Header Available; Continue (T or F):  ',$)
053   FORMAT (L1)
054   FORMAT (' --- Write Aborted ---')
55    FORMAT (' <:Header Not Updated For This Scan:>')
56    FORMAT (' <><>Do You Want To Update Header? ',$)
57    FORMAT (A1)
88    FORMAT (' ::File Already Exists:: ',6A1)
89    FORMAT (' <><> Do You Wish To [ Abort,Rename, or Continue]? ',$)
C
      IF (KT.EQ.0) TYPE 051   !NO DATA CHECK
      IF (KT.EQ.0) RETURN
C
      IF (.NOT.HDR) TYPE 052          !IF NO HDR
      IF (.NOT.HDR) ACCEPT 053,ANS
      IF (.NOT.ANS) RETURN

C
      IF (ISTYP.GE.20) GO TO 888 !HAS HDR BEEN UPDATED
      TYPE 55
      TYPE 56
```

```
        ACCEPT 57,I
        IF (I.EQ.'Y') CALL SUBHDR
C
888     TYPE 004,LINE1  !TYPE HDR
        TYPE 004,LINE2
        TYPE 004,PARAM
        TYPE 001
C
997     IF(L-2)111,222,333
111     TYPE 009
        ACCEPT 002,FNMT
        IF (FNMT(2).EQ.32) TYPE 054     ! IF BLANK THEN RETURN
        IF (FNMT(2).EQ.32) RETURN
        FNMT(31)=0
        OPEN (UNIT=13,NAME=FNMT,TYPE='OLD',FORM='UNFORMATTED',
     &  READONLY,ERR=1098)
        CLOSE (UNIT=13,DISP='SAVE')
        TYPE 88,(FNMT(III),III=1,6)
        TYPE 89
        ACCEPT 57,I
        IF (I.EQ.'R') GO TO 997
        IF (I.EQ.'A') GO TO 1077
1098    OPEN (UNIT=13,NAME=FNMT,TYPE='NEW',FORM='UNFORMATTED',
     &  ERR=1000)
        GOTO 200
C
222     TYPE 008
        ACCEPT 002,FNMB
        IF(FNMB(2).EQ.32) TYPE 054
        IF(FNMB(2).EQ.32) RETURN
        FNMB(31)=0
        OPEN (UNIT=13,NAME=FNMB,TYPE='OLD',FORM='UNFORMATTED',
     &  READONLY,ERR=2000)
        CLOSE (UNIT=13,DISP='SAVE')
        TYPE 88,(FNMB(III),III=1,6)
        TYPE 89
        ACCEPT 57,I
        IF(I.EQ.'R') GOTO 997
        IF(I.EQ.'A')GOTO 1077
2000    OPEN (UNIT=13,NAME=FNMB,TYPE='NEW',FORM='UNFORMATTED',
     &  ERR=1000)
        GOTO 200
C
333     TYPE 007
        ACCEPT 002,FNMS
        IF(FNMS(2).EQ.32) TYPE 054
        IF(FNMS(2).EQ.32) RETURN
        FNMS(31)=0
        OPEN (UNIT=13,NAME=FNMS,TYPE='OLD',FORM='UNFORMATTED',
     &  READONLY,ERR=3000)
```

```
      CLOSE (UNIT=13,DISP='SAVE')
      TYPE 88,(FNMS(III),III=1,6)
      TYPE 89
      ACCEPT 57,I
      IF(I.EQ.'R')GOTO 997
      IF(I.EQ.'A')GOTO 1077
3000  OPEN (UNIT=13,NAME=FNMB,TYPE='NEW',FORM='UNFORMATTED',
     & ERR=1000)
C
200   WRITE (13) LINE1
      WRITE (13) LINE2
      WRITE (13) PARAM
C
      DO 210 K1=1,KT
        WRITE (13) ARRAY(K1,1),ARRAY(K1,2)
210   CONTINUE
C
      WRITE (13) ARRAY(3603,1),ARRAY(3603,2)
      WRITE (13) ARRAY(3604,1),ARRAY(3604,2)
C
      CLOSE (UNIT=13,DISP='SAVE')
C
1077  RETURN
C
1000  TYPE 050
      GOTO 200
      END
```

```
C*************************************************************************

C*************  THIS SUBROUTINE PRINTS ANY DESIRED PORTION  ************

C*************  OF A DATA FILE ON THE "CRT" SCREEN .        ************

C*************************************************************************

C
      SUBROUTINE PRNTDF(UT,UB)
      VIRTUAL UT(3604,2),UB(3604,2)
      INTEGER XX
C
      COMMON/FLAGS/MT,MB,MS,MN,MST
      COMMON/WQY/XX
C
66    FORMAT(/,' YOU HAVE NOT READ A DATA FILE ,SO CANNOT',/
     +  ,' EXECUTE YOUR COMMAND .',)
77    FORMAT(A2)
78    FORMAT(/,' ILLEGAL FILE SPECIFICATION ,TRY AGAIN .',)
87    FORMAT(/,'    ENTER TYPE OF FILE  :  ',$)
C
85    TYPE 86
86    FORMAT(/,'    TARGET FILE       :  TF',/
     +  ,'    BACKGROUND FILE   :  BF',/
     +  ,'    SHPERE FILE       :  SF',/
     +  ,'    CALIBRATED FILE   :  CF',/
     +  ,'    < FILE "Y" >      :  YF',/
     +  ,'    < FILE "X" >      :  XF',/)
      TYPE 87
      ACCEPT 77,XX
      IF(XX.EQ.'TF')GOTO 25
      IF(XX.EQ.'BF')GOTO 30
      IF(XX.EQ.'SF')GOTO 30
      IF(XX.EQ.'CF')GOTO 25
      IF(XX.EQ.'YF')GOTO 25
      IF(XX.EQ.'XF')GOTO 30
      TYPE 78
      GOTO 85
25    IF(MT.EQ.1)GOTO 26
      TYPE 66
      GOTO 99
26    CALL PNTOUT(UT)
      GOTO 99
30    IF(MB.EQ.1)GOTO 36
      IF(MS.EQ.1)GOTO 36
      TYPE 66
      GOTO 99
36    CALL PNTOUT(UB)
99    RETURN
      END
```

```
C*************************************************************************

C************  THIS SUBROUTINE DOES THE COMPUTATIONS FOR      **********

C************  PRINTING THE DATA SPECIFIED BY PRNTDF(ARRAY)   **********

C*************************************************************************

C
      SUBROUTINE PNTOUT(ARRAY)
      VIRTUAL ARRAY(3604,2)
C
      COMMON/LISFRQ/LOFQ,IUPFQ,INCRE
      COMMON/LOOP/JK,IK
      COMMON/INPTS1/ALPHA1,ALPHA2
      COMMON/ASPCTA/ALPHA,DELTA1
C
60    FORMAT(F7.0)
50    FORMAT(7X,F8.2,7X,F8.2,6X,F8.2)
70    FORMAT(/,'   ******************************************',/
     +  ,'       ASPECT ANGLE     AMPLITUDE      PHASE ',/
     +  ,'        (degrees)         (dB"s)     (degrees)',/
     +  ,'      **************  ************  ************ ',/)

10    FORMAT(/,' ENTER THE INITIAL VALUE OF ASPECT ANGLE ',/
     +  ,'       ALPHA1  =  ',$)
20    FORMAT(/,' ENTER THE FINAL VALUE OF ASPECT ANGLE ',/
     +  ,'       ALPHA2  =  ',$)
C
C  ALPHA1 & ALPHA2 DEFINES THE INTERVAL OF ASPECT ANGLE SPECIFYING THE
C  PORTION OF A DATA FILE TO BE PRINTED OUT .
40    TYPE 10
      READ(5,60,ERR=661)ALPHA1
      IF(ALPHA1.LT.LOFQ)GOTO 2003
      GOTO 45
2003  TYPE*,'   (!) INITIAL ASPECT ANGLE IS SMALL (!)'
      GOTO 40
45    TYPE 20
      READ(5,60,ERR=663)ALPHA2
      IF(ALPHA2.GT.IUPFQ)GOTO 2004
      GOTO 2005
2004  TYPE*,'   (!) FINAL ASPECT ANGLE IS LARGE (!)'
2005  DELTA1=INCRE/100.
      IK=((ALPHA1-LOFQ)/DELTA1)+1
      JK=((ALPHA2-LOFQ)/DELTA1)+1
C  IK & JK ARE THE CORRESPONDING VALUES OF ARRAY SUBSCRIPT .
      TYPE 70
      DO 27 I=IK,JK
      ALPHA=((I-1)*DELTA1)+LOFQ
      WRITE(7,50)ALPHA,ARRAY(I,1),ARRAY(I,2)
```

```
27      CONTINUE
        RETURN
661     TYPE 662
662     FORMAT(/,' (*) ILLEGAL FORMAT ,TRY AGAIN .(*)',)
        GOTO 40
663     TYPE 662
        GOTO 45
        RETURN
        END
```

```
C*************************************************************************
C************** THIS SUBROUTINE EDITS THE HEADER OF A DATA FILE  *******
C*************************************************************************
C
      SUBROUTINE EDTHDF
C
      COMMON/FLAGS/MT,MB,MS,MST
C
      IF((MT.EQ.1).OR.(MB.EQ.1).OR.(MS.EQ.1))GOTO 20
      TYPE 10
10    FORMAT(/,'  YOU HAVE NOT READ A DATA FILE .',/
     +  ,'  SO YOUR COMMAND IS NOT EXECUTABLE .',/)
      GOTO 40
20    CALL SUBHDR
40    RETURN
      END
```

```
      SUBROUTINE SUBHDR
C
C     ****************************************
C     THIS ROUTINE GENERATES THE HEADER
C     ****************************************
C
C
C
C
C
C
C     VARIABLES
C
      BYTE CMD(70),CUR(60),LINE1(60),LINE2(60),PARAM(60)
      BYTE WORD1(30),WORD2(30)
      DOUBLE PRECISION SCNTYP
      COMMON /LISFRQ/ LOFQ,IUPFQ,INCRE
      COMMON /MHDR/ LINE1,LINE2,PARAM,HDR
      COMMON /FILE/ ITS,KT,ISTYP,IFLG
      COMMON /SCNFCT/ DELTA,START
      LOGICAL HDR
C
C     FORMATS
C
1     FORMAT (' ')
3     FORMAT (' Line #',I1,':  ',$)
10    FORMAT (70A1)
11    FORMAT (Q,31A1)
12    FORMAT ('0',60A1)
13    FORMAT (' --- Terminator (#) was not used ---')
14    FORMAT (' ',60A1)
16    FORMAT (' -Line #?',$)
17    FORMAT (I1)
18    FORMAT (' >>Replace:  ',$)
20    FORMAT (20A1)
22    FORMAT (' >>With:  ',$)
24    FORMAT (' --- No Such Word ---')
30    FORMAT ('NL=')
31    FORMAT (I4)
32    FORMAT (' FF=')
33    FORMAT (I5)
34    FORMAT (' IN=')
35    FORMAT (' TYP=')
36    FORMAT (' >>Replace: ',$)
38    FORMAT (' >>With: ',$)
40    FORMAT (30A1)
50    FORMAT ('NA=')

52    FORMAT (' FA=')
53    FORMAT (I5)
54    FORMAT (' IN=')
```

```
55      FORMAT (I2)
C
        IF (ISTYP.LT.20) ISTYP=ISTYP+20 !ISTYP TELL OF HDR UPDATE
C
C       SET 3RD LINE PARAMETERS FOR FRQ SCN
C
        IF (MOD(ISTYP,2).NE.0) GO TO 353
        ENCODE (3,030,PARAM(1))
        ENCODE (4,031,PARAM(4)) KT
        ENCODE (4,032,PARAM(8))
        ENCODE (5,033,PARAM(12)) LOFQ
        ENCODE (4,034,PARAM(17))
        ENCODE (5,033,PARAM(21)) INCRE
        ENCODE (5,35,PARAM(26))
        ENCODE (2,55,PARAM(31)) ISTYP
        ENCODE (1,1,PARAM(33))
        GO TO 354
C
C       SET 3RD LINE PARAMETERS FOR AZIMUTH SCN
C
353     IF (MOD(ISTYP,2).NE.1) GO TO 354
        ENCODE (3,050,PARAM(1))
        ENCODE (4,031,PARAM(4)) KT
        ENCODE (4,052,PARAM(8))
        ENCODE (5,053,PARAM(12)) IFIX(START)
        ENCODE (4,054,PARAM(17))
        ENCODE (5,053,PARAM(21)) INCRE
        ENCODE (5,35,PARAM(26))
        ENCODE (2,55,PARAM(31)) ISTYP
        ENCODE (1,1,PARAM(33))
C
C       --- CHARACTER EDITOR ---
C
354     IF (HDR) GOTO 100
        DO 8 K=1,3
          TYPE 3,K
          IF (K .EQ. 1) ACCEPT 10,LINE1
          IF (K .EQ. 2) ACCEPT 10,LINE2
          IF (K .EQ. 3) ACCEPT 11,IZH2,(PARAM(K1),K1=34,IZH2+34)
8       CONTINUE
        HDR=.TRUE.
C
100     NPOS=1
        TYPE 12,LINE1
        TYPE 14,LINE2
        TYPE 14,PARAM

105     TYPE 16
C
        READ (5,17,ERR=105) L
```

```
        IF (L.EQ.0) RETURN
        IF (L.LT.1.OR.L.GT.3) GOTO 105
C
        DO 110 K=1,60
          IF (L.EQ.1) CUR(K)=LINE1(K)
          IF (L.EQ.2) CUR(K)=LINE2(K)
          IF (L.EQ.3) CUR(K)=PARAM(K)
110     CONTINUE
C
        TYPE 12,CUR
        ACCEPT 10,CMD
        IF (CMD(1).EQ.'/') GOTO 599
C
120     DO 200 N=1,70
          IF (CMD(N) .EQ. '#') GOTO 210
          IF (CMD(N).EQ.'%') GOTO 150
          IF (CMD(N).EQ.'©') GOTO 300
          IF (CMD(N).EQ.' ') NPOS=NPOS+1
          IF (CMD(N).EQ.' ') GOTO 200
          GOTO 170
C
150       DO 160 K=NPOS,59
            CUR(K)=CUR(K+1)
160       CONTINUE
          CUR(60)=32
          GOTO 200
C
170       DO 180 K=1,60-NPOS
            CUR(61-K)=CUR(60-K)
180       CONTINUE
          CUR(NPOS)=CMD(N)
          NPOS=NPOS+1
C
200     CONTINUE
C
210     CONTINUE
        DO 220 K=1,60
          IF (L.EQ.1) LINE1(K)=CUR(K)
          IF (L.EQ.2) LINE2(K)=CUR(K)
          IF (L.EQ.3) PARAM(K)=CUR(K)
220     CONTINUE
        GOTO 100
C
300     N=N+1
        IF (N .LE. 61) GOTO 305
        TYPE 13

        GOTO 100
305     IF (CMD(N) .EQ. '#') GOTO 210
C
```

```
        DO 310 K=1,60-NPOS
          CUR(61-K)=CUR(60-K)
310     CONTINUE
C
        CUR(NPOS)=CMD(N)
        NPOS=NPOS+1
        GOTO 300
C
C       --- WORD SEARCH ---
C
599     DO 598 K1=1,60
          IF (L.EQ.1) CUR(K1)=LINE1(K1)
          IF (L.EQ.2) CUR(K1)=LINE2(K1)
          IF (L.EQ.3) CUR(K1)=PARAM(K1)
598     CONTINUE
C
597     TYPE 36
        ACCEPT 40,WORD1
C
        TYPE 38
        ACCEPT 40,WORD2
C
        DO 590 K1=1,60
          CMD(K1)=32
590     CONTINUE
C
        LW1=0
        LW2=0
        DO 591 K1=1,30
          IF (WORD1(K1).NE.32) LW1=LW1+1
          IF (WORD2(K1).NE.32) LW2=LW2+1
591     CONTINUE
C
        KW=1
        DO 600 K1=1,60
          IF (KW.EQ.1) IPT=K1
          IF (CUR(K1).NE.WORD1(KW)) KW=1
          IF (CUR(K1).NE.WORD1(KW)) GOTO 600
          IF (K1.EQ.1.OR.KW.EQ.1) GOTO 601
          IF (CUR(K1-1).NE.WORD1(KW-1)) KW=1
          IF (CUR(K1-1).NE.WORD1(KW-1)) GOTO 600
601       CONTINUE
          CMD(K1)=37
          KW=KW+1
          IF (KW.GT.LW1) GOTO 604
600     CONTINUE

C
604     KW=0
        DO 605 K1=1,60
```

```
          IF (K1.LT.IPT) CMD(K1)=32
          IF (K1.GE.IPT.AND.CMD(K1).NE.37) KW=KW+1
          IF (K1.GE.IPT.AND.CMD(K1).NE.37) CMD(K1)=WORD2(KW)
          IF (KW.EQ.LW2) GOTO 606
605       CONTINUE
C
606       GOTO 120
          END
```

```
C C C C C C C C C C C C C C C C C C
        SUBROUTINE PLOTS(AR,ITME,ISETS)
        INTEGER ARPNT
        COMMON /PVTPAR/ AMN,AMX,PMN,PMX,SL,SH
        COMMON /FILE/ NSMP(2),INTRV(2),NBACK,NUMFIL,ARPNT
        VIRTUAL AR(1001,2,2),ITME(100,3,2)
        BYTE TXT(6)     !MESSAGE BUFFER
        DIMENSION       VAL(3)
C
C]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
C       QUICK PLOT
C]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
C
C
C
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C       This routine initializes the VT125 for plotting
C       by drawing the brackets and labels
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C
C       formats
10      FORMAT (' ','No timming trigger data')
C
C
C       EXECUTABLE
C
C
        CALL FRAME
C
C       initialize plot limit pointers
        SL=0.
        SH=NSMP(ARPNT)
C
        CALL NUMBER (SL,SH)
C
C
C
        ISTEP=0
        IF (ISETS.EQ.1) ISTEP=NSMP(ARPNT)/60.
        IF (ISTEP.LE.1) ISTEP=1
        CALL PLACEP(VAL,0)
C
        DO 200 I=1,NSMP(ARPNT),ISTEP
C
C       abort option detection
C
        IQUIT=-1
        CALL CRDETR(IQUIT)
        IF (IQUIT.EQ.1) RETURN
C
```

```
        VAL(1)=AR(I,1,ARPNT)
        VAL(2)=AR(I,2,ARPNT)
        VAL(3)=I-1
        CALL PLACEP(VAL,1)
200     CONTINUE
C
        CALL DRMOV(0,245)
        IF (INTRV(ARPNT).NE.0)GO TO 300
        TYPE 10
        RETURN
300     DO 310 I=1,INTRV(ARPNT)
        IF (ITME(I,2,ARPNT).GE.0) ITME(I,2,ARPNT)=1
        IF (ITME(I,2,ARPNT).LT.0) ITME(I,2,ARPNT)=-1
        IF (I.EQ.1) GOTO 208
        CALL DRDRW(IFIX((ITME(I,3,ARPNT)-1)*600./NSMP(ARPNT)),
     &  245+ITME(I-1,2,ARPNT)*15)
208     CALL DRDRW(IFIX((ITME(I,3,ARPNT)-1)*600./NSMP(ARPNT)),
     &  245+ITME(I,2,ARPNT)*15)
C
310     CONTINUE
        RETURN
        END
```

```
      SUBROUTINE DRBFRT(COMS, LEN)
C
C  SUBROUTINE "DRBFRT" (FOR 'DEC ReGIS BuFfer Routine') BUFFERS A
C  SERIES OF
C  ReGIS COMMANDS FOR EVENTUAL FLUSHING. THIS ROUTINE PREFIXES EACH
C  BUFFER
C  FUL OF ReGIS COMMANDS WITH THE  <ESC>Pp  'PLACE DEVICE IN GRAPHICS
C  MODE'
C  ReGIS COMMAND.  WHEN THE OUTPUT BUFFER IS FLUSHED AFTER A COMPLETE
C  ReGIS
C  COMMAND-STRING HAS BEEN RECIEVED AND THE BUFFER IS FULL,  <ESC>® IS
C  SENT
C  TO TAKE THE ReGIS DEVICE OUT OF GRAPHICS MODE.     IN THE PRESENT
C  OUTPUT
C  BUFFER CONFIGURATION,  THE SIZE OF THE ENTIRE BUFFER IS 512 BYTES.
C  PART
C  OF  THE  BUFFER IS HELD IN RESERVE TO RECIEVE CHARACTERS WHILE THE FI
C  FIRST

C  PART  IS  BEING  FLUSHED.   THE PARAMETER 'EOBPT' HOLDS THE VALUE
C  OF THE
C  SIZE OF THIS "OVERFLOW" BUFFER. ITS PRESENT VALUE IS 64.
C
      LOGICAL*1 INIT
      INTEGER*2 LEN                              ! LENGTH OF INCOMING
C                                                  COMS

      BYTE COMS(LEN)                             ! INCOMING COMMAND-
C                                                STRING
      INTEGER*2 COMIDX                           ! COMS   ARRAY INDEX
      INTEGER*2 BUFIDX                           ! BUFFER ARRAY INDEX
      INTEGER*2 EOBPT                            ! SIZE OF OVERFLOW
C                                                BUFFER
      INTEGER*2 MODE            ! BUFFERING MODE (0=NONE, 1=BUFFERING
C                               ENABLED)
      INTEGER*2 BUFLEN                           ! LENGTH OF OUTPUT
C                                                BUFFER
      BYTE BUFFER(512)                           ! OUTPUT BUFFER IS
C                                                BUFLEN LONG
C
      COMMON /BLOCK1/MODE,BUFIDX                 ! COMMON BLOCK1
      COMMON /BLOCK2/BUFFER                      ! COMMON BLOCK2
C
      DATA EOBPT / 64 /          ! LENGTH OF OVERFLOW BUFFER IS 64 BYTES
      DATA BUFLEN / 512 /        ! LENGTH OF OUTPUT BUFFER IS 512 BYTES
      DATA INIT / .FALSE. /     ! INITIALIZE FLAG --ASSUMED TO BE
C                               UNINITIZLIZED
C
      BUFFER(1) = 27                             ! "<ESC>"
      BUFFER(2) = 80                             ! "P"
      BUFFER(3) = 112                            ! "p"
```

```
C
      IF (MODE .NE. 1)                          ! IF BUFFERING IS NOT
C                                               DESIRED
     +  GOTO 100    ! PLACE ReGIS TERMINAL COMMANDS AROUND THE COMMAND-
C                   STRING
C
      IF (INIT) GOTO 10                 ! IF ROUTINE HAS NOT BEEN
C                                       INITIALIZED
        INIT = .TRUE.                           ! INITIALIZE THE ROUTINE
        BUFIDX = 4      ! NOW POINTS PAST "<ESC>Pp" AT BEGINNING
C                       BUFFER
C     ENDIF
C
10    COMIDX = 1                        ! NOW POINTS TO BEGINNING OF
C                                       COMS
20    IF (BUFIDX.GE.((BUFLEN-EOBPT)+1)) GOTO 30 ! DO WHILE ...
        BUFFER(BUFIDX) = COMS(COMIDX)   ! BUFFER GETS 1 CHARACTER FROM
C                                       COMS
        BUFIDX = BUFIDX+1                       ! INCREMENT BUFFER
C                                               POINTER
        COMIDX = COMIDX+1                       ! INCREMENT COMS POINTER
        IF ((COMIDX-1) .GE. LEN) GOTO 999  ! WHEN COMPLETELY DUMPED,
C                                          RETURN
      GOTO 20                                   ! ENDWHILE
C
30    INIT = .FALSE.                    ! FLAG ROUTINE FOR RE-
C                                       INITIALIZATION
C
40    IF ((COMIDX-1) .GE. LEN)GOTO 50 ! DO WHILE COMS IS NOT COMPLETELY
C                                     DUMPED
        BUFFER(BUFIDX) = COMS(COMIDX)   ! CONTINUE TO FILL OVERFLOW
C                                       BUFFER
        BUFIDX = BUFIDX+1                       ! INCREMENT BUFFER
C                                               POINTER
        COMIDX = COMIDX+1                       ! INCREMENT COMS POINTER
      GOTO 40                                   ! ENDWHILE
C
50    BUFFER(BUFIDX) = 27       ! AT END OF THE SERIES OF COMMANDS,
C                               APPEND
      BUFIDX = BUFIDX+1         ! INCREMENT BUFFER POINTER
      BUFFER(BUFIDX) = 92       ! ASCII "<ESC>®"
C
      CALL DRFLSH               ! FLUSH THE OUTPUT BUFFER
C

      GOTO 999                          ! SKIP PAST NOBUFFERING ROUTINE
C
C----NOBUFFERING ROUTINE--INSERTS "<ESC>Pp" BEFORE A REGIS COMMAND-
C    STRING, AND
C----"<ESC>®"  AFTER  THE  COMMAND-STRING.   THESE ReGIS STRINGS PUT THE
C    ReGIS
```

```
C----DEVICE INTO AND OUT OF GRAPHICS MODE, RESPECTIVELY.
C
100   BUFIDX = 3          ! WILL POINT PAST "<ESC>Pp" AT BEGINNING OF
C                         BUFFER
C
      DO 110 COMIDX = 1,LEN      ! PLACE COMS IN BUFFER, BEHIND "<ESC>Pp"
        BUFFER(BUFIDX+COMIDX) = COMS(COMIDX)    !          ...
110   CONTINUE                                  ! NEXT COMIDX
      BUFIDX = LEN+4              ! POINTS TO NEXT CHARACTER POSITION IN
C                                 BUFFER
      BUFFER(BUFIDX) = 27         ! NEXT CHARACTER IN BUFFER IS "<ESC>"
      BUFIDX = BUFIDX+1           ! INCREMENT BUFFER POINTER
      BUFFER(BUFIDX) = 92         ! NEXT CHARACTER IN BUFFER IS "®"
C
      CALL DRFLSH                                ! FLUSH THE OUTPUT
R                                                BUFFER
C
999   RETURN
      END
```

```
      SUBROUTINE DRBUFR(DRMODE)
C
C  "DRBUFR" SETS THE BUFFERING MODE FLAG FOR OUTPUT BUFFERING CONTROL
C  DRMODE:  0=NOBUFFERING, 1=BUFFERING ENABLED, 2=FLUSH OUTPUT BUFFER
C
      INTEGER*2 DRMODE                          ! PASSED BUFFERING MODE
      INTEGER*2 BUFIDX                          ! BUFFER INDEX POINTER
      INTEGER*2 MODE            ! BUFFERING MODE--ASSUME BUFFERING
C                               ENABLED
      LOGICAL*1 LTFLSH                          ! "LET FLUSH" FLAG
C
      INTEGER*2 BUFLEN                          ! LENGTH OF OUTPUT
C                                               BUFFER
      BYTE BUFFER(512)                          ! OUTPUT BUFFER IS
C                                               BUFLEN LONG
      COMMON /BLOCK1/MODE,BUFIDX                ! COMMON BLOCK1
      COMMON /BLOCK2/BUFFER                     ! COMMON BLOCK2
      DATA BUFLEN / 512 /               ! LENGTH OF OUTPUT BUFFER IS 512
C                                       BYTES
C
      IF(DRMODE .NE. 2) MODE = DRMODE           ! RESET MODE
      IF (DRMODE .NE. 0.AND.DRMODE .NE. 1       ! IF AN INVALID MODE IS
C                                               GIVEN
     + .AND.DRMODE .NE. 2) MODE=1 ! ASSUME A MODE OF 1 (= BUFFERING ENAB
                                  ENABLED)
C
      IF (MODE .EQ. 1) LTFLSH = .TRUE.  ! ALLOW OUTPUT BUFFER FLUSHING
      IF (MODE .EQ. 0) LTFLSH = .FALSE. ! DISALLOW OUTPUT BUFFER FLUSHIN
C                                       FLUSHING
C
      IF (DRMODE .NE. 2) RETURN         ! IF NO FLUSH REQUESTED, RETURN
      IF (.NOT. LTFLSH) RETURN          ! IF BUFFER FLUSHING IS ALLOWED
        BUFFER(BUFIDX) = 27             ! NEXT BYTE IN BUFFER GETS
C                                       "<ESC>"
        BUFIDX = BUFIDX+1               ! POINT TO NEXT AVAILABLE BYTE
        BUFFER(BUFIDX) = 92             ! BYTE GETS "®"

        CALL DRFLSH                     ! FLUSH THE ENTIRE OUTPUT BUFFER
C     ENDIF                             ! ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE DRCOLR(IC)
C
C  SUBROUTINE DRCOLR ASSIGNS A LINE COLOR THAT A ReGIS DEVICE WILL
C  DRAW IN
C
      BYTE IC                              ! COLOR (SEE TABLE BELOW)
      BYTE COMS(5)                         ! COMMAND-STRING IS 5 BYTES LONG
C
C--COLOR: 0=DARK, 1=BLUE, 2=RED, 3=MAGENTA, 4=GREEN, 5=CYAN, 6=YELLOW,
C  7 = WHITE
C
      DATA COMS / 87, 40, 73,              ! ASCII "W(I
     +            0,                       ! ASCII NULL CHARACTER
     +            41 /                     ! ASCII ")"
      ENCODE(1,100,COMS(4)) IC             ! CONVERT CONTENTS OF IC TO
C                                          ASCII
C
C-----COMMAND-STRING NOW LOOKS LIKE: W(Ii)
C-----WHERE i IS THE SELECTED WRITING COLOR [INTENSITY].
C
      CALL DRBFRT(COMS,5)  ! WRITE COMS TO ReGIS DEVICE VIA THE ReGIS BU
C                          BUFFER
100   FORMAT(I1)
C
      RETURN
      END
```

```
      SUBROUTINE DRDRW(IX,IY)
C
C  SUBROUTINE DRDRW DRAWS A LINE (VECTOR) FROM THE PRESENT COORDINATE
C  POSITION
C  OF THE GRAPHICS
C  CURSOR ON A ReGIS DEVICE TO  THE  COORDINATE  POSITION  GIVEN  BY THE
C  PARAMETERS IX AND IY.
C  THE GRAPHICS CURSOR IS LEFT AT POSITION (IX,IY).
C
      INTEGER*4 IX                        ! X-COORDINATE
      INTEGER*4 IY                        ! Y-COORDINATE
      BYTE COMS(12)                       ! COMMAND-STRING IS 12 BYTES
C                                         LONG
      DATA COMS / 86, 91,                         ! ASCII "V["
     +            0, 0, 0, 0,                     ! ASCII NULL CHARACTERS
     +            44,                             ! ASCII ","
     +            0, 0, 0, 0,                     ! ASCII NULL CHARACTERS
     +            93 /                            ! ASCII "]"
      ENCODE(4,100,COMS(3)) IX            ! CONVERT CONTENTS OF IX TO
C                                         ASCII
      ENCODE(4,100,COMS(8)) IY            ! CONVERT CONTENTS OF IY TO ASCI
C                                         ASCII
C
C-----COMMAND-STRING NOW LOOKS LIKE: V[<IX>,<IY>]

C-----WHERE <IX> AND <IY> ARE THE ASCII-CONVERTED CONTENTS OF IX AND IY,
C-----RESPECTIVELY;
C-----PADDED ON THE LEFT WITH BLANKS IF NECESSARY (LENGTH OF 4).
C
      CALL DRBFRT(COMS,12)  ! WRITE COMS TO ReGIS DEVICE VIA THE ReGIS
C                           BUFFER
100   FORMAT(I4)
C
      RETURN
      END
```

```
      SUBROUTINE DREGIS(COMS,LEN)
C
C  SUBROUTINE DREGIS WRITES A ReGIS COMMAND-STRING TO A ReGIS DEVICE.
C
      INTEGER*4 LEN                                  ! LENGTH OF ReGIS
C                                                    COMMAND
      BYTE COMS(LEN)                                 ! COMMAND-STRING IS
C                                                    'LEN' LONG
C
      CALL DRBFRT(COMS,LEN) ! WRITE COMS TO ReGIS DEVICE VIA THE ReGIS
C                           BUFFER
C
      RETURN
      END
```

```
      SUBROUTINE DRFLSH
C
C  SUBROUTINE "DRFLSH" FLUSHES THE ReGIS OUTPUT BUFFER, GIVEN AN INDEX
C  THAT
C  POINTS TO THE LAST CHARACTER TO BE READ INTO THE BUFFER (BUFIDX).
C
      INTEGER*2 MODE                              ! BUFFERING MODE
      INTEGER*2 BUFIDX                            ! BUFFER ARRAY INDEX
      INTEGER*2 BUFLEN                            ! LENGTH OF OUTPUT
C                                                 BUFFER
      BYTE BUFFER(512)                            ! OUTPUT BUFFER IS
C                                                 BUFLEN LONG
C
      COMMON /BLOCK1/MODE,BUFIDX                  ! COMMON BLOCK1
      COMMON /BLOCK2/BUFFER                       ! COMMON BLOCK2
C
      DATA BUFLEN / 512 /              ! LENGTH OF OUTPUT BUFFER IS 512
C                                      BYTES
C
      CALL QIO(BUFFER,BUFIDX) ! FLUSH THE ReGIS OUTPUT BUFFER
      BUFIDX = 4                                  ! RESET BUFFER INDEX
C                                                 POINTER
C
      RETURN
      END
```

```
      SUBROUTINE DRINIT
C
C   SUBROUTINE  DRINIT  PERFORMS  THE  FOLLOWING  ON  A  ReGIS  DEVICE :
C  ERASES THE SCREEN,  SETS THE "PEN" COLOR TO WHITE, HOMES THE GRAPHICS
C  CURSOR TO  (0,0)  [ULC],  SETS THE SCREEN COLOR TO "DARK",  POSITIONS
C  THE  TEXT CURSOR  AT  THE BOTTOM OF THE SCREEN,  CLEARS TEXT FROM THE

C  SCREEN, SETS THE DEVICE TO ANSI MODE, ASSUMES OUTPUT BUFFERING, ReGIS
C  TEXT SIZE OF 1, HEIGHT OF 2, DIRECTION OF 0 DEGREES, AND NO ITALICS.
C
      BYTE COMS(52)                          ! COMMAND-STRING IS 52 BYTES
C                                            LONG
C
C-----DATA FOR COMS IS ASCII CHARACTER CODES
C
      DATA COMS / 27, 60,          ! "<ESC><"    IS ReGIS FOR "ENTER ANSI
C                                  MODE

     +            27, 91, 50, 48, 66,          ! "<ESC>[20B"
     +            27, 91, 50, 74,              ! "<ESC>[2J"
     +            27, 80, 112,                 ! "<ESC>Pp"
     +            32, 83, 40, 69, 41,          ! " S(E)"
     +            32, 87, 40, 73, 55, 41,      ! " W(I7)"
     +            32, 80, 91, 48, 44, 48, 93,  ! " P[0,0]"
     +            32, 83, 40, 73, 48, 41,      ! " S(I0)"
     +            32, 84, 40, 72, 50, 41,      ! " T(H2)"
     +            32, 84, 40, 73, 48, 41,      ! " T(I0)"
     +            27, 92 /                     ! "<ESC>\"
C
      CALL QIO(COMS,52)                      ! WRITE COMS TO ReGIS DEVICE
      CALL DRBUFR(0)                         ! NO OUTPUT BUFFERING
      CALL DRSTXT(1,0)                       ! ASSUME TEXT HEIGHT OF 1,
C                                            DIRECTION 0
      CALL DRBUFR(1)                         ! ASSUME OUTPUT BUFFERING IS
C                                            DESIRED
C
      RETURN
      END
```

```
      SUBROUTINE DRMOV(IX,IY)
C
C  SUBROUTINE DRMOV  POSITIONS THE  ReGIS  DEVICE GRAPHICS CURSOR
C  AT THE COORDINATES GIVEN BY THE PARAMETERS GIVEN BY IX AND IY.
C
      INTEGER*4 IX                          ! X-COORDINATE
      INTEGER*4 IY                          ! Y-COORDINATE
      BYTE COMS(12)                         ! COMMAND-STRING IS 12 BYTES
C                                           LONG
      DATA COMS / 80, 91,                           ! ASCII "P["
     +            0, 0, 0, 0,                       ! ASCII NULL CHARACTERS
     +            44,                               ! ASCII ","
     +            0, 0, 0, 0,                       ! ASCII NULL CHARACTERS
     +            93 /                              ! ASCII "]"
      ENCODE (4,100,COMS(3)) IX             ! CONVERT CONTENTS OF IX TO
C                                           ASCII
      ENCODE (4,100,COMS(8)) IY             ! CONVERT CONTENTS OF IY TO
C                                           ASCII
C
C-----COMMAND-STRING NOW LOOKS LIKE: P[<IX>,<IY>]
C-----WHERE <IX> AND <IY> ARE THE ASCII-CONVERTED CONTENTS OF IX AND IY,
C-----RESPECTIVELY;
C-----PADDED ON THE LEFT WITH BLANKS IF NECESSARY (LENGTH OF 4).
C
      CALL DRBFRT(COMS,12)  ! WRITE COMS TO ReGIS DEVICE VIA THE ReGIS
C                           BUFFER

100   FORMAT(I4)
C
      RETURN
      END
```

```
      SUBROUTINE DRTERM
C
C  SUBROUTINE "DRTERM" ERASES  A ReGIS DEVICE'S GRAPHICS MEMORY AND
C  FLUSHES
C  THE ReGIS OUTPUT BUFFER.
C
      INTEGER*2 BUFIDX                  ! BUFFER ARRAY INDEX
      INTEGER*2 BUFLEN                  ! LENGTH OF OUTPUT BUFFER
      INTEGER*2 MODE                    ! BUFFERING MODE
      BYTE COMS(9)                      ! COMMAND-STRING IS 9 BYTES LONG
      BYTE BUFFER(512)                  ! OUTPUT BUFFER IS BUFLEN LONG
C
      COMMON /BLOCK1/MODE,BUFIDX        ! COMMON BLOCK1
      COMMON /BLOCK2/BUFFER             ! COMMON BLOCK2
C
      DATA BUFLEN / 512 /               ! LENGTH OF OUTPUT BUFFER IS 512
C                                      BYTES
      DATA COMS / 27, 80, 112,          ! "<ESC>Pp"
     +            83, 40, 69, 41,       ! "S(E)"
     +            27, 92 /              ! "<ESC>\"
C
C
      CALL QIO(COMS,9)                  ! WRITE COMS TO ReGIS DEVICE
      BUFIDX = 4                        ! RESET BUFFER INDEX POINTER
C
      RETURN
      END
```

```
        SUBROUTINE DRSTXT(SIZE, ANGLE)
C
C       "DRSTXT" SETS THE SIZE AND DIRECTION THAT ReGIS TEXT WILL BE
C       DRAWN IN. THE SIZE MAY BE ANY INTEGER BETWEEN 0 AND 16,  AND
C       THE DRAWING ANGLE MAY BE ANY INTEGER VALUE BETWEEN -360  AND
C       360  DEGREES.  ReGIS  WILL  TAKE THE ANGLE TO BE THE NEAREST
C       MULTIPLE OF 45 DEGREES.
C
        INTEGER*2 SIZE                          ! HOLDS DESIRED TEXT
C                                               SIZE
        INTEGER*2 ANGLE                         ! HOLDS DESIRED DIRECTIO
C                                               DIRECTION
        BYTE COMS(20)                           ! HOLDS ReGIS COMMAND-
C                                               STRING
C
        DATA COMS / 84, 40, 68,                 ! ASCII "T(D"
     +               0,  0,  0,  0,             ! ASCII NULL CHARACTERS
     +              41,                         ! ")"
     +              40, 83,                     ! "(S"
     +               0,  0,                     ! ASCII NULL CHARACTERS
     +              41,                         ! ASCII ")"
     +              40, 68,                     ! "(D"
     +               0,  0,  0,  0,             ! ASCII NULL CHARACTERS
     +              41 /                        ! ASCII ")"
C
C
        IF ((SIZE .LT. 0) .OR. (SIZE .GT. 16)) SIZE = 1
        IF ((ANGLE .LT. -360) .OR. (ANGLE .GT. 360)) ANGLE = 0
C
C-----CONVERT CONTENTS OF "SIZE" TO ASCII AND INSERT IN COMS
        ENCODE (2, 100, COMS(11)) SIZE
C
C-----CONVERT CONTENTS OF "ANGLE" TO ASCII AND INSERT IN COMS
        ENCODE (4, 200, COMS(4))  ANGLE
        ENCODE (4, 200, COMS(16)) ANGLE
C
C-----COMS NOW LOOKS LIKE: T(Ddirection)(Ssize)(Ddirection)
C
        CALL DRBFRT(COMS,20)                    ! WRITE TO ReGIS DEVICE
C
100     FORMAT(I2)
200     FORMAT(I4)
C
        RETURN
        END
```

```
      SUBROUTINE DRTEXT(TEXT, LEN)
C
C     "DRTEXT" PRINTS ReGIS-GENERATED TEXT ON A ReGIS DEVICE.  THE
C     MAXIMUM LENGTH OF "TEXT" IS 85 CHARACTERS.
C
      INTEGER*2 LEN                          ! LENGTH OF INCOMING
C                                            STRING
      BYTE TEXT(1)                           ! HOLDS INCOMING STRING
      BYTE COMS(88)                          ! HOLDS OUT-GOING ReGIS
C                                            STRING
C
      COMS(1) = 84                           ! ASCII "T"
      COMS(2) = 39                           ! ASCII "'"
C
      IF (LEN .LE. 0) RETURN                 ! NO TEXT PRINTED
      IF (LEN .GT. 85) LEN = 85              ! MAX. LENGTH IS 85
C
      DO 10 I=3,LEN+2                        ! PACK TEXT INTO COMS
         COMS(I) = TEXT(I-2)                 !         ...
10    CONTINUE                               ! NEXT I
C
      COMS(LEN+3) = 39                       ! LAST CHAR. IN COMS IS
C                                            "'"
C
      CALL DRBFRT(COMS,LEN+3)                ! WRITE COMS TO ReGIS
C                                            DEVICE
C
      RETURN

      END
```

```
      SUBROUTINE QIO(COMS,LEN)
C
C  SUBROUTINE  QIO  WRITES  THE  CONTENTS  OF  A  CHARACTER BUFFER TO
C  AN  I/O  CHANNEL,  GIVEN THE ADDRESS OF THE BUFFER AND ITS LENGTH.
C  THE BUFFER MUST BE A BYTE ARRAY. THIS ROUTINE IS WRITTEN IN VAX-11
C  FORTRAN.
C
      INTEGER*4 LEN                          ! HOLDS LENGTH OF COMMAND-STRING
      BYTE COMS(1)                           ! HOLDS VALUE OF COMMAND-STRING

C
      COMS(LEN+1)=128
      CALL PRINT(COMS)
      RETURN
      END
```

```
      SUBROUTINE FRAME
C     Executable
C
      CALL DREGIS (20H;S(A[0,479][767,0]);,20)        !Set up vt125
C
C     Draw frames
C
      DO 100 I=1,5              !Frame loop>
      I1=(I-1)*50+275           !Upper horizonal
      CALL DRMOV(0,I1)
      CALL DRDRW(600,I1)
      CALL DRMOV(0,I1-250)      !Lower horizonal
      CALL DRDRW(600,I1-250)
      I1=(I-1)*150
      CALL DRMOV(I1,275)        !Vertical
      CALL DRDRW(I1,475)
      CALL DRMOV(I1,25)
      CALL DRDRW(I1,225)
100   CONTINUE
C
      CALL DRMOV (615,35)       !Label the vertical axies
      CALL DRTEXT ('AMP',3)
      CALL DRMOV (615,310)
      CALL DRTEXT ('PHA',3)
C
      RETURN
      END
```

```
        SUBROUTINE PLACEP(VAL,ITYPE)
C
C"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
C       This routine is a point  plotting routine
C       that requires data to be sent through VAL

C       and an initialization call with VAL undetermined
C       and ITYPE=0. ITYPE equal 1 for plotting.
C"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
C
C       common and data definition
C
        INTEGER ARPNT,ALAST,PLAST,SLAST
        DIMENSION VAL(3)
        COMMON /PVTPAR/ AMN,AMX,PMN,PMX,SL,SH
        COMMON /PARPLT/ S1L,S1H,ALAST,PLAST,SLAST
     &  ,AFCTOR,PFCTOR,XFCTOR
C
C
C               formats
10      FORMAT (' ','**** invalid mode in placep routine')
C'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
C'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
C       executable code
C
        I=ITYPE+1
        GO TO (100,200) I       !execute type of input
C
        TYPE 10                 !input type error
        RETURN                  !reutrn /nothing done/
C
100     ALAST=25                !initialize values to beginning of plot
        PLAST=275               !amp,pha,xaxis
        SLAST=0
        AFCTOR=200./(AMX-AMN)   !scal factors set for all axis
        PFCTOR=200./(PMX-PMN)
        XFCTOR=600./(S1H-S1L)
        RETURN                  !initializing done
C
200     CALL DRMOV(SLAST,ALAST)         !move to last amplitude position
        IX=(VAL(3)-S1L)*XFCTOR          !calculate next xaxis position
        ALAST=(VAL(1)-AMN)*AFCTOR+25    !calculate next amp position
        CALL DRDRW (IX,ALAST)           !draw to new point
        CALL DRMOV (SLAST,PLAST)        !move to last phase position
        PLAST=(VAL(2)-PMN)*PFCTOR+275   !find new phase point
        SLAST=IX                        !set to new xaxis position
        CALL DRDRW(SLAST,PLAST)         !draw to new phase point
        RETURN                  !!done with single point drawing
C
        END
```

```
      SUBROUTINE CLRCRT
C
C"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
C       This routine clears the CRT screen and the
C       plotting buffers.
C
C"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
      CALL DRINIT
      CALL DRTERM
      RETURN
      END
```

```
      SUBROUTINE NUMBER(LOW,HIGH)
C
C*************************************************************
C     This routine lables the x-axis for plots on the CRT
C     The low and high values input are real*4 and are used
C     as end points for the labeling and calculation of
C     placing of points.
C
C
C*************************************************************
C
C     common data and data definition
C
      REAL LOW,HIGH
      COMMON /PARPLT/ S1L,S1H
      COMMON /PVTPAR/ AMN,AMX,PMN,PMX
      BYTE TXT(10)
C
C           formats
10    FORMAT(F5.2)
11    FORMAT (F6.1)
12    FORMAT(F5.0)
C
C     executable code
C
      S1L=LOW
      S1H=HIGH
      DELTAS=(S1H-S1L)/4.
      DELTAA=(AMX-AMN)/4.
      DELTAP=(PMX-PMN)/4.
      DO 100 I=1,5
      ENCODE (5,12,TXT) S1L+(I-1)*DELTAS
      CALL DRMOV((I-1)*145,18)
      CALL DRTEXT (TXT,5)
      CALL DRMOV((I-1)*145,268)
      CALL DRTEXT (TXT,5)
      ENCODE (6,11,TXT) AMN+(I-1)*DELTAA
      CALL DRMOV(610,25+(I-1)*50)
      CALL DRTEXT (TXT,6)
      ENCODE (6,11,TXT) PMN+(I-1)*DELTAP
      CALL DRMOV(610,275+(I-1)*50)
      CALL DRTEXT (TXT,6)
100   CONTINUE
      RETURN
      END
```

```
C*************************************************************************
C***********  THIS SUBROUTINE PREFORMS THE SUBTRACTION OF TWO  *********
C***********  DATA FILES .                                     *********
C*************************************************************************
C
      SUBROUTINE SUBTDF(UT,UB)
      VIRTUAL UT(3604,2),UB(3604,2)
      INTEGER S
C
      COMMON/FILE/ITS1,KT
      COMMON/ANSFR/S
      COMMON/FLAGS/MT,MB,MS,MN,MST
      COMMON/KRSB/IJK
C
      IJK=0
      TYPE 10
10    FORMAT(/,' SUBTRACTION WILL BE PREFORMED IN THE FOLLOWING',/
     +  ,' ORDER : < FILE "Y" > = < FILE "Y" > - < FILE "X" > ',/
     +  ,' THE RESULT WILL BE STORED IN FILE "Y" .',)
      TYPE 11
11    FORMAT(/,5X,'  (*) DEFINE FILE "X"  (*) ',/)
      CALL SUBREA(UB,4)
      IF(IJK.EQ.1)RETURN
      IJK=0
      TYPE 12
12    FORMAT(/,5X,'  (*)  DEFINE FILE "Y"  (*) ',/)
      CALL SUBREA(UT,5)
      IF(IJK.EQ.1)RETURN
      IJK=0
      FCT=57.2958
C   THIS DO-LOOP COMPUTES THE DIFFERENCE .
      DO 35 K=1,KT
      RTY=10.0**(UT(K,1)/20.0)
      RBX=10.0**(UB(K,1)/20.0)
      PTY=UT(K,2)/FCT
      PBX=UB(K,2)/FCT
      RL=(RTY*COS(PTY))-(RBX*COS(PBX))
      CX=(RTY*SIN(PTY))-(RBX*SIN(PBX))
      A=(RL**2.0)+(CX**2.0)
      IF(A.EQ.0)GOTO 3004
      GOTO 3005
3004  TYPE*,' X = Y  AT DATA POINT  #',K
      UT(K,1)=UT(K-1,1)
      UT(K,2)=UT(K-1,2)
      GOTO 35
3005  UT(K,1)=10.0*(ALOG10(A))
      UT(K,2)=FCT*(ATAN2(CX,RL))
```

```
35      CONTINUE

C
864     FORMAT(10A1)
        TYPE 864,' ',1799
        TYPE 20
20      FORMAT(/,' DO YOU WANT TO CHANGE THE HEADER ?',/
     +  ,' TYPE "Y" FOR YES ,PUSH RETURN FOR NO .',/)
        ACCEPT 29,S
29      FORMAT(A1)
        IF(S.EQ.'Y')GOTO 48
        GOTO 50
48      CALL SUBHDR
50      TYPE 51
51      FORMAT(/,' DO YOU WANT TO WRITE THIS FILE ?',/
     +  ,' TYPE "Y" FOR YES ,PUSH RETURN FOR NO ',/)
        ACCEPT 29,S
        IF(S.EQ.'Y')GOTO 46
        MT=1
        RETURN
46      CALL SUBWRI(UT,1)
        MT=1
        RETURN
        END
```

```
C*************************************************************************

C**************   THIS SUBROUTINE CALIBRATES A DATA FILE   *************

C*************************************************************************

C
        SUBROUTINE CALBDF(UT,UB)
        VIRTUAL UT(3604,2),UB(3604,2)
        INTEGER S
        COMPLEX TT,BB,SS,TMB,SMB,RR
        BYTE LINE1(60),LINE2(60),PARAM(60)
        BYTE FNMT(31),FNMB(31),FNMS(31)
        COMMON/FLG/FLAG
        COMMON/FILE/ITS1,KT,ISTYP,IFLG
        COMMON/LISFRQ/LOFQ,IUPFQ,INCRE
        COMMON/MHDR/LINE1,LINE2,PARAM,HDR
        COMMON/MWRI/FNMT,FNMB,FNMS
        COMMON/SFIND/STRG,KSTRG,KCHR
        DOUBLE PRECISION STR(25)
        LOGICAL HDR
C
        COMMON/TARGET/RLT,CXT,RLCLB,CXCLB,RDCLB
        COMMON/BKDSP/RLB,CXB,RLS,CXS,FCTR
        COMMON/CCXX/TT,BB,SS,RR
        COMMON/FLAGS/MT,MB,MS,MN,MST,MSD
        COMMON/PAR/ASAF,ASAI,ASAINC
        COMMON/MULTF/AE
        COMMON/KRSB/IJK
C

C   (*) FIRST STEP IS TO OPEN SHPHERE FILE .
C
002     FORMAT (31A1)
003     FORMAT (' ',60A1)
004     FORMAT (I4)
005     FORMAT (I5)
010     FORMAT (' <><> ENTER SPHERE FILE NAME : ',$)
050     FORMAT (' OPEN ERROR -- FILE DOES NOT EXIST.')
051     FORMAT (' DECODE ERROR -- LINE COUNTER "KT".')
052     FORMAT (' --- READ ABORTED ---')
88      FORMAT (I2)
C
C
        IF(MSD.EQ.1)GOTO 102    !CHECK IF AUTOMATIC DATA READ IS SET.
        TYPE 010                !GET FILE NAME
        ACCEPT 002,FNMS
C
102     IF (FNMS(2).EQ.32)TYPE 052
        IF (FNMS(2).EQ.32)RETURN
C
```

```
C       OPEN FILE
C
        OPEN (UNIT=14,NAME=FNMS,TYPE='OLD',FORM='UNFORMATTED',
     &  READONLY,ERR=1000)
C
C               READ HEADER
        READ (14) LINE1
        READ (14) LINE2
        READ (14) PARAM
        TYPE 003,LINE1
        TYPE 003,LINE2
        TYPE 003,PARAM
        HDR=.TRUE.
C
C       DECODE HEADER
C
        IF (PARAM(8).EQ.'F')GOTO 995
        DECODE (4,004,PARAM(4),ERR=1001)KT
        DECODE (5,005,PARAM(12),ERR=1001)LOFQ
        DECODE (5,005,PARAM(21),ERR=1001)INCRE
        DECODE (2,88,PARAM(31),ERR=1008) ISTYP
        GOTO 388
995     DECODE (4,2004,PARAM(4),ERR=1001) KT
2004    FORMAT (I3)
        DECODE (5,005,PARAM(11),ERR=1001) LOFQ
        DECODE (5,005,PARAM(20),ERR=1001) INCRE
        ISTYP=0
C
C


C       SET BAKAVE FLAGS AND TYPES [SCN TYPE]
C
388     IFLG=0
        IF (MOD(ISTYP,10).NE.3) GOTO 389
        IFLG=2
389     CNE=INCRE/100.0
        IUPFQ=IFIX(LOFQ+(KT-1)*CNE)
        GOTO 91
C
        FMIN=LOFQ
        FMAX=IUPFQ
1000    TYPE 050
        RETURN
1001    TYPE 051
        RETURN
1008    TYPE *,'OLD FILE TYPE'
        ISTYP=-1
        GOTO 388
C
C**********    NOW THE SPHERE FILE HAS BEEN OPENED  ******************
C
```

```
91      FCTR=57.2958
C   THE ABOVE NUMBER IS A CONVERSION FACTOR ,DEGREES/RADIANS.
C
555     FORMAT(2X,7A1)
23      FORMAT(A1)
        IF(MB.EQ.1)GOTO 26
        CALL SUBREA(UB,2)
        IF(IJK.EQ.1)RETURN      !CHECKING IF BAKGND. FILE EXISTS.
        GOTO 25
26      TYPE 555,FNMB
        TYPE 27
        ACCEPT 23,S
        IF(S.EQ.'Y')GOTO 25
        CALL SUBREA(UB,2)
        IF(IJK.EQ.1)RETURN
C********   NOW THE DESIRED BACKGROUND FILE IS READ IN *****************

C
25      CALL SUBREA(UT,1)
        IF(IJK.EQ.1)RETURN
C
C****** NOW PROGRAM MADE SURE THAT THE DESIRED TARGET FILE IS READ IN **

C
27      FORMAT(/,' IS A BACKGROUND FILE ALREADY IN EXISTANCE.',/
     +  ,' IF YOU WANT THIS FILE, TYPE "Y".',/
     +  ,' TO CHANGE THIS FILE ,PUSH RETURN .'//,$)
C*************************************************************************

C
        IF(MSD.EQ.1)GOTO 37     !CHECKS IF AUTOMATIC READ FLAG IS SET.
36      IF(MN.EQ.1)GOTO 30      !CHECKS IF AN OLD VALUE EXACT FILE
C                               EXISTS
34      CALL READNM
        GOTO 37
30      TYPE 31
31      FORMAT(/,' A VALUE FOR THE MULTIPLYING FACTOR ALREADY EXISTS',/)
        WRITE(7,32)AE
32      FORMAT(/,5X,' MULTIPLYING FACTOR = ',F8.2)
        TYPE 33
33      FORMAT(/,' IF YOU WANT THIS VALUE ,TYPE "Y" .',/
     +  ,' IF YOU WANT TO CHANGE IT ,PUSH RETURN .',//,$)
        ACCEPT 23,S
        IF(S.EQ.'Y')GOTO 37
        GOTO 34
C
C****** NOW THE PROGRAM MADE SURE THAT THE DESIRED VALUE FOR THE    ****

C****** MULTIPLYING FACTOR IS READ IN.                              ****

C
```

```
37      ASAINC=INCRE/100.0
        M=((IUPFQ-LOFQ)/(ASAINC))+1      !TOTAL NUMBER OF POINTS.
C
C*************  NOW PROGRAM IS READY TO PREFORM CALIBRATION ************

C
        DO 19 I=1,M
        READ(14) US1,US2
        UT(I,1)=10.0**(UT(I,1)/20.0)
        UB(I,1)=10.0**(UB(I,1)/20.0)
        US1=10.0**(US1/20.0)
        UT(I,2)=(UT(I,2))/FCTR
        UB(I,2)=(UB(I,2))/FCTR
        US2=US2/FCTR
        RLT=(UT(I,1))*(COS(UT(I,2)))
        RLB=(UB(I,1))*(COS(UB(I,2)))
        RLS=US1*(COS(US2))
        CXT=(UT(I,1))*(SIN(UT(I,2)))
        CXB=(UB(I,1))*(SIN(UB(I,2)))
        CXS=US1*(SIN(US2))
        TT=CMPLX(RLT,CXT)
        BB=CMPLX(RLB,CXB)
        SS=CMPLX(RLS,CXS)
        TMB=TT-BB
        SMB=SS-BB
C
C ADJUSTING DATA WHEN DIVISION BY ZERO OCCURS .
C
        IF(REAL(TMB))440,441,440
441     TYPE*,' TAR = BKGND  AT DATA POINT #',I
        UT(I,1)=UT(I-1,1)
        UT(I,2)=UT(I-1,2)
        GOTO 19

440     IF(REAL(SMB))442,443,442
443     TYPE*,' SPHR = BKGND  AT DATA POINT #',I
        UT(I,1)=UT(I-1,1)
        UT(I,2)=UT(I-1,2)
        GOTO 19
442     RR=TMB/SMB
C
        RLCLB=REAL(RR)
        CXCLB=AIMAG(RR)
        RDCLB=(RLCLB**2.0)+(CXCLB**2.0)
        UT(I,1)=10.0*(ALOG10(RDCLB))+AE
        UT(I,2)=FCTR*(ATAN2(CXCLB,RLCLB))
19      CONTINUE
        CLOSE (UNIT=14,DISP='SAVE')
C
C***************   NOW ARRAY UT CONTAINS THE CALIBRATED FILE ***********
```

```
C
864     FORMAT(10A1)
        TYPE 864,' ',1799
        MT=1
        MB=0
        MS=0
        TYPE 109
109     FORMAT(/,' DO YOU WANT TO CHANGE THE HEADER ON THE ',/
     + ,' CALIBRATED FILE ? IF YES TYPE "Y" ,IF NOT PUSH RETURN .')
        ACCEPT 23,S
        IF(S.EQ.'Y')GOTO 114
        GOTO 113
114     CALL EDTHDF
113     TYPE 112
112     FORMAT(//,' DO YOU WANT TO WRITE THE CALIBRATED FILE ?',/
     +  ,' IF YES TYPE "Y" ,IF NOT PUSH RETURN .',/)
        ACCEPT 23,S
        IF(S.EQ.'Y')GOTO 115
        GOTO 500
115     CALL SUBWRI(UT,1)
C
500     RETURN
        END
```

```
C*************************************************************************

C***************  THIS SUBROUTINE PLOTS A DATA FILE  ******************

C*************************************************************************

C
      SUBROUTINE PLOTDF(UT,UB)
      BYTE FNMT(31),FNMB(31),FNMS(31)
      VIRTUAL UT(3604,2),UB(3604,2)
      REAL VAL(3)
      INTEGER R,FK
C

      COMMON/EEE/R,FK
      COMMON/FLAGS/MT,MB,MS,MN,MST
      COMMON/TOTNM/M,I,J,MMM,IY
      COMMON/DDD/VAL,DELTA
      COMMON/MWRI/FNMT,FNMB,FNMS
      COMMON/LISFRQ/LOFQ,IUPFQ,INCRE
      COMMON/PVTPAR/AMN,AMX,PMN,PMX
      COMMON/VARR/BGSTA,SMSTA,BGSTP,SMSTP
      COMMON/KRSB/IJK
C
100   TYPE 101
101   FORMAT(/,'    TARGET FILE       :  TF ',/
     +,'    BACKGROUND FILE   :  BF ',/
     +,'    SPHERE FILE       :  SF ',/
     +,'    CALIBRATED FILE   :  CF ',/,)
      TYPE 102
102   FORMAT(2X,'  ENTER TYPE OF FILE  :  ',$)
      ACCEPT 103,FK
103   FORMAT(A2)
      IF(FK.EQ.'TF')GOTO 11
      IF(FK.EQ.'BF')GOTO 22
      IF(FK.EQ.'SF')GOTO 22
      IF(FK.EQ.'CF')GOTO 11
      TYPE 104
104   FORMAT(/,' (*) ERROR IN ENTERING FILE SPECIFICATION .')
      GOTO 100
C * * * * * * * * * * * * * * * * * * * * * * * *
11    IF(MT.EQ.1)GOTO 1030            !CHECKING IF FILE EXISTS
      GOTO 28
1030  IF(IUPFQ.EQ.0)GOTO 28           !CHECKING IF DATA EXISTS
      WRITE(7,707)FNMT
707   FORMAT(/,' FILE NAME :',1X,31A1)
      TYPE 15
15    FORMAT(/,'  THE ABOVE IS A DATA FILE IN EXISTANCE.',/
     +,' IF YOU WANT TO PLOT THAT FILE ,THEN TYPE "Y" .',/
     +,' IF YOU WANT TO PLOT ANOTHER ONE ,TYPE ANY OTHER ',/
     +,' LETTER .',//,$)
```

```
        ACCEPT 18,R
18      FORMAT(A1)
        IF(R.EQ.'Y')GOTO 19
C  THE USER STILL HAS THE OPTION OF PLOTTING ANOTHER TARGET FILE
C  DEPENDING ON THE VALUE OF R .
        CALL SUBREA(UT,1)
        IF(IJK.EQ.1)GOTO 28              !CHECK AGAIN IF FILE EXISTS.
19      DELTA=INCRE/100.0
        M=((IUPFQ-LOFQ)/(DELTA))+1
        CALL MINMAX(UT)
        IY=1
C  IY=1  =>  PLOT A TARGET FILE .

        GOTO 363
C * * * * * * * * * * * * * * * * * * * * * * * * *
C  CHEKING IF THERE ACTUALY IS A DATA FILE IN VIRTUAL MEMORY .
C
22      IF((MB.EQ.1).OR.(IUPFQ.NE.0))GOTO 14    !CHECK IF SPHERE FILE
C                                                EXISTS.
        IF((MS.EQ.1).OR.(IUPFQ.NE.0))GOTO 140   !CHECK IF BAKGND. FILE
C                                                EXISTS.
        GOTO 28
14      WRITE(7,707)FNMB
        TYPE 15
        ACCEPT 18,R
        IF(R.EQ.'Y')GOTO 33
        CALL SUBREA(UB,2)
        IF(IJK.EQ.1)GOTO 28              !CHECK IF DATA FILE EXIST
        GOTO 33
140     WRITE(7,777)FNMS
777     FORMAT(/,' FILE NAME : ',31A1)
        TYPE 15
C
C  THE USER STILL HAS A CHANCE TO PLOT ANOTHER SPHERE OR BACKGROUND
C  FILE.
C
        ACCEPT 18,R
        IF(R.EQ.'Y')GOTO 33
        CALL SUBREA(UB,3)
        IF(IJK.EQ.1)GOTO 28
33      DELTA=INCRE/100.0        !ACTUAL INCREMENT IN ASPECT ANGLE.
        M=((IUPFQ-LOFQ)/(DELTA))+1       !TOTAL NUMBER OF POINTS.
        CALL MINMAX(UB)
        IY=2
C
C   IY=2  =>  PLOT A SPHERE OR BACKGROUND FILE .
C * * * * * * * * * * * * * * * * * * * * * * * * *
C
363     IF(MST.EQ.1)GOTO 125     !CHECKING IF A SCALE HAS BEEN SET.
        TYPE 364
```

```
364     FORMAT(/,'  DO YOU DESIRE TO SET YOUR OWN SCALE ?',/
     +  ,'  IF YES ,TYPE "Y". IF NO TYPE ANY OTHER LETTER ',/,$)
        ACCEPT 18,R
        IF(R.EQ.'Y')GOTO 366
        GOTO 170
366     CALL SETSCL
        GOTO 625
C
C************  SETTING AN ARBITRARY SCALE  ***********
170     IF(BGSTA.LT.0)GOTO 171
        AMX=10.0*(INT(BGSTA/10.0))+10.0
        GOTO 172
171     AMX=10.0*(INT(BGSTA/10.0))
172     IF(SMSTA.LT.0)GOTO 173
        AMN=10.0*(INT(SMSTA/10.0))

        GOTO 174
173     AMN=10.0*(INT(SMSTA/10.0))-10.0
174     IF(BGSTP.LT.0)GOTO 175
        PMX=10.0*(INT(BGSTP/10.0))+10.0
        GOTO 176
175     PMX=10.0*(INT(BGSTP/10.0))
176     IF(SMSTP.LT.0)GOTO 177
        PMN=10.0*(INT(SMSTP/10.0))
        GOTO 125
177     PMN=10.0*(INT(SMSTP/10.0))-10.0
C
C*****************************************************
C
125     TYPE 105
105     FORMAT(/,' *  THE PRESENT SCALE IS  *')
        WRITE(7,710)AMX
710     FORMAT(/,' MAXIMUM AMPLITUDE = ',F8.2)
        WRITE(7,711)AMN
711     FORMAT(' MINIMUM AMPLITUDE = ',F8.2)
        WRITE(7,712)PMX
712     FORMAT(' MAXIMUM PHASE     = ',F8.2)
        WRITE(7,713)PMN
713     FORMAT(' MINIMUM PHASE     = ',F8.2)
C
C
        TYPE*,'  '
        TYPE*,' IF YOU WANT THE ABOVE SCALE PUSH RETURN  .'
        TYPE*,' IF YOU WANT TO CHANGE THE SCALE TYPE "C" .'
        TYPE*,' IF YOU WANT THE MODULO-10 SCALE TYPE "T" .'
        ACCEPT 18,R
        IF(R.EQ.'C')CALL SETSCL
        IF(R.EQ.'T')GOTO 170
C
```

```
C . . . . . SETTING OF HORIZONTAL SCALE  . . . . . .
C
1006    FORMAT(F8.2)
625     TYPE 1001
1001    FORMAT(/,' ENTER INITIAL VALUE OF ASPECT ANGLE : ',$)
        ACCEPT*,X1
        IF(X1.LT.LOFQ)GOTO 1003
        GOTO 1004
1003    TYPE*,' INITIAL VALUE OF ASPECT ANGLE IS SMALL .'
        GOTO 625
1004    TYPE 1002
1002    FORMAT(/,' ENTER FINAL VALUE OF ASPECT ANGLE : ',$)
        ACCEPT*,X2
        N1=((X1-LOFQ)/DELTA)+1
        N2=((X2-LOFQ)/DELTA)+1
C . . . . . .
C
        IF(IY-1)126,126,111
126     CALL CLRCRT
        CALL DRBUFR(0)
        CALL FRAME
        CALL NUMBER(X1,X2)
        CALL PLACEP(VAL,0)
C
C   THIS DO LOOP COMPUTES THE LOCATION OF THE DOT ON THE SCREEN .
C   ITTINR() IS A PDP-11 SOFTWARE PACKAGE ,IT ALLOWS THE USER IN THIS
C   CASE TO INTRRUPT THE PLOTTING PROCEDURE UPON PRESSING CARRIGE
C   RETURN.
C   AND ALSO GOING BACK TO COMMAND MODE WHEN <CR> IS FOLLOWED BY "Q" .
C
110     DO 20 I=N1,N2
        IF(UT(I,1).GT.AMX) VAL(1)=AMX
        IF(UT(I,1).GT.AMX)GOTO 2050
        IF(UT(I,1).LT.AMN) VAL(1)=AMN
        IF(UT(I,1).LT.AMN)GOTO 2050
        VAL(1)=UT(I,1)
2050    IF(UT(I,2).GT.PMX) VAL(2)=PMX
        IF(UT(I,2).GT.PMX)GOTO 2060
        IF(UT(I,2).LT.PMN) VAL(2)=PMN
        IF(UT(I,2).LT.PMN)GOTO 2060
        VAL(2)=UT(I,2)
2060    VAL(3)=X1+((I-N1)*DELTA)
        CALL PLACEP(VAL,1)
        ICH=ITTINR()
        IF(ICH.LT.0)GOTO 20
        ACCEPT 18,R
        IF(R.EQ.'Q')GOTO 507
        GOTO 20
507     MMM=I
        I=N2
20      CONTINUE
```

```
        GOTO 90
C****************************************
111     CALL CLRCRT
        CALL DRBUFR(0)
        CALL FRAME
        CALL NUMBER(X1,X2)
        CALL PLACEP(VAL,0)
C
        DO 27 I=N1,N2
        IF(UB(I,1).GT.AMX) VAL(1)=AMX
        IF(UB(I,1).GT.AMX)GOTO 1007
        IF(UB(I,1).LT.AMN) VAL(1)=AMN
        IF(UB(I,1).LT.AMN)GOTO 1007
        VAL(1)=UB(I,1)
1007    IF(UB(I,2).GT.PMX) VAL(2)=PMX

        IF(UB(I,2).GT.PMX)GOTO 1008
        IF(UB(I,2).LT.PMN) VAL(2)=PMN
        IF(UB(I,2).LT.PMN)GOTO 1008
        VAL(2)=UB(I,2)
1008    VAL(3)=X1+((I-N1)*DELTA)
        CALL PLACEP(VAL,1)
        ICH=ITTINR()
        IF(ICH.LT.0)GOTO 27
        ACCEPT 18,R
        IF(R.EQ.'Q')GOTO 510
        GOTO  27
510     MMM=I
        I=N2
27      CONTINUE
        GOTO 90
C*****************************************
C
28      TYPE 29
29      FORMAT(/,'  YOU HAVE NOT READ IN A FILE TO PLOT .',/
     +  ,'  SO ,CANNOT EXECUTE PLOT SUBROUTINE .',//,$)
90      RETURN
        END
```

```
C*************************************************************************

C*************  THIS SUBROUTINE CLEARS THE "CRT" SCREEN  *****> *********

C*************************************************************************

C
C
        CALL CLRCRT
        RETURN
        END
```

```
C*************************************************************************
C*********   THIS SUBROUTINE COMPUTES MAXIMUM & MINIMUM VALUES   *******
C*********   OF THE AMPLITUDE AND PHASE IN A DATA FILE .         *******
C*************************************************************************
C
      SUBROUTINE MINMAX(ARRAY)
      VIRTUAL ARRAY(3604,2)
      REAL UX(2),UM(2)
      COMMON/FILE/ITS1,KT
      COMMON/VARR/BGSTA,SMSTA,BGSTP,SMSTP
C
C   THE VARAIBLES :
C      BGSTA = MAXIMUM AMPLITUDE .
C      SMSTA = MINIMUM AMPLITUDE .
C      BGSTP = MAXIMUM PHASE .
C      SMSTP = MINIMUM PHASE .
C   SET INITIAL VALUES .
C
      BGSTA=-99999.0

      BGSTP=-99999.0
      SMSTA=999999.0
      SMSTP=999999.0
C
      DO 30 I=1,KT
      BGSTA=AMAX1(BGSTA,ARRAY(I,1))
      BGSTP=AMAX1(BGSTP,ARRAY(I,2))
      SMSTA=AMIN1(SMSTA,ARRAY(I,1))
30    SMSTP=AMIN1(SMSTP,ARRAY(I,2))
C
      WRITE(7,734)BGSTA
734   FORMAT(/,'  MAXIMUM AMPLITUDE = ',F7.2)
      WRITE(7,735)SMSTA
735   FORMAT('  MINIMUM AMPLITUDE = ',F7.2)
      WRITE(7,736)BGSTP
736   FORMAT('  MAXIMUN PHASE     = ',F7.2)
      WRITE(7,737)SMSTP
737   FORMAT('  MINIMUM PHASE     = ',F7.2)
      RETURN
      END
```

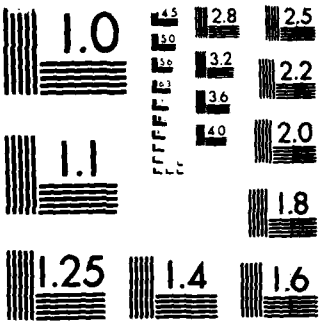
```
C*************************************************************************
C************  THIS SUBROUTINE SETS THE PLOTTING SCALES  ***************
C*************************************************************************
C
      SUBROUTINE SETSCL
C
      INTEGER R
      COMMON/FLAGS/MT,MB,MS,MN,MST
      COMMON/PVTPAR/AMN,AMX,PMN,PMX
      COMMON/ANSWER/R
C
175   FORMAT(/,' (*) ILLEGAL FORMAT ,TRY AGAIN .(*)',)
12    FORMAT(A1)
55    FORMAT(F7.0)
66    TYPE 10
10    FORMAT(/,'  ENTER THE MAXIMUM AMPLITUDE  ',/
     +  ,'       AMX = ',$)
      READ(5,55,ERR=200)AMX
19    TYPE 20
20    FORMAT(/,'  ENTER THE MINIMUM AMPLITUDE  ',/
     +  ,'       AMN = ',$)
      READ(5,55,ERR=210)AMN
29    TYPE 30
30    FORMAT(/,'  ENTER THE MAXIMUM PHASE ANGLE ',/
     +  ,'       PMX = ',$)
      READ(5,55,ERR=220)PMX
39    TYPE 40
40    FORMAT(/,'  ENTER THE MINIMUM PHASE ANGLE ',/
     +  ,'       PMN = ',$)
      READ(5,55,ERR=230)PMN
C
50    FORMAT(/,'  HAVE YOU MADE A TYPING ERROR ?',/
     +  ,'  IF YES TYPE "C" ,IF NOT PUSH RETURN .',/)
C
      GOTO 123
200   TYPE 175
      GOTO 66
210   TYPE 175
      GOTO 19
220   TYPE 175
      GOTO 29
230   TYPE 175
      GOTO 39
C
123   TYPE 80
80    FORMAT(/,' MAXIMUM AMP. = ',$)
      WRITE(7,55)AMX
      TYPE 82
82    FORMAT(,' MINIMUM AMP. = ',$)
      WRITE(7,55)AMN
      TYPE 84
```

```
84      FORMAT(,' MAXIMUM PHS. = ',$)
        WRITE(7,55)PMX
        TYPE 86
86      FORMAT(' MINIMUM PHS. = ',$)
        WRITE(7,55)PMN
C
        TYPE 50
        ACCEPT 12,R
        IF(R.EQ.'C')GOTO 66
        MST=1
        RETURN
        END
```

```
C*************************************************************************
C*********  THIS SUBROUTINE SETS THE DATA FILE NAMES INVOLVED IN THE  **
C*********  CALIBRATION PROCEDURE IN A BUFFER .                       **
C*** *********************************************************************
C
        SUBROUTINE SETDTA
        BYTE FNMT(31),FNMB(31),FNMS(31)
        COMMON/MWRI/FNMT,FNMB,FNMS
        COMMON/MULTF/AE
        COMMON/FLAGS/MT,MB,MS,MN,MST,MSD
C
45      FORMAT(31A1)
        TYPE 10
10      FORMAT(/,' TARGET FILE NAME      : (1)',/
     +  ,' BACKGROUND FILE NAME  : (2) ',/
     +  ,' SPHERE FILE NAME      : (3) ',/
     +  ,' EXACT FILE VALUE      : (4) ',/
     +  ,' TO LIST FILE NAMES    : (5) ',/
     +  ,' TO EXIT ,PUSH RETURN .')
        TYPE 15
15      FORMAT(/,' TYPE THE NUMBER IN ( ) TO CHOOSE OPTION .')
77      TYPE 46
46      FORMAT(/,' OPTION ?',$)
        ACCEPT 13,I
13      FORMAT(I1)
        IF(I.EQ.1)GOTO 11
        IF(I.EQ.2)GOTO 22
        IF(I.EQ.3)GOTO 33
        IF(I.EQ.4)GOTO 44
        IF(I.EQ.5)GOTO 55
        IF(I.EQ.0)GOTO 190
        GOTO 77
11      TYPE 12
12      FORMAT(/,' (*) ENTER TARGET FILE NAME  : ',$)
        ACCEPT 45,FNMT
        MSD=1
        GOTO 77
22      TYPE 23
23      FORMAT(/,' (*) ENTER BACKGROUND FILE NAME  : ',$)
        ACCEPT 45,FNMB
        MSD=1
        GOTO 77
33      TYPE 34
34      FORMAT(/,' (*) ENTER SPHERE FILE NAME  : ',$)
        ACCEPT 45,FNMS
        MSD=1
        GOTO 77
```

AD-A162 527 ASPECT SCAN USER'S PROGRAM FOR RCS MEASUREMENT[illegible]
STATE UNIV COLUMBUS ELECTROSCIENCE LAB
A JALLOUL ET AL MAY 84 ESL-714190-7 N00014-82-K-0037
F/G 9/2 NL

UNCLASSIFIED



```
44      CALL READNM
        MSD=1           !SET FLAG
        GOTO 77
55      WRITE(7,91)FNMT
        WRITE(7,92)FNMB
        WRITE(7,93)FNMS
        WRITE(7,94)AE
91      FORMAT(/,' TARGET     : ',31A1)
92      FORMAT(/,' BACKGROUND : ',31A1)
93      FORMAT(/,' SPHERE     : ',31A1)
94      FORMAT(/,' MULT. FCTR.: ',F8.2)
        GOTO 77
190     RETURN
        END
```

```
C*************************************************************************
C************  THIS SUBROUTINE SETS THE FLAGS FOR THE PROGRAM  **********
C*************************************************************************
C
      SUBROUTINE SETFLG
C
      COMMON/FLAGS/MT,MB,MS,MN,MST,MSD
      COMMON/ARBFG/IF
C
303   FORMAT(/,' (*) ILLEGAL FORMAT ,TRY AGAIN .(*)',)
25    FORMAT(I1)
      WRITE(7,245)MT,MB,MS,MN,MST,MSD
      DO 29 I=1,6
12    TYPE 10,I
10    FORMAT(/,' ENTER FLAG #(',I1,') = ',$)
      READ(5,25,ERR=298)IF
      GOTO(11,22,33,44,55,66)I
11    MT=IF
      GOTO 29
22    MB=IF
      GOTO 29
33    MS=IF
      GOTO 29
44    MN=IF
      GOTO 29
55    MST=IF
      GOTO 29
66    MSD=IF
      GOTO 29
29    CONTINUE
      WRITE(7,245)MT,MB,MS,MN,MST,MSD
245   FORMAT(/,' FLAGS = ',6(I1),/)
      RETURN
298   TYPE 303
      RETURN
      END
```

```
C*************************************************************************
C********  THIS SUBROUTINE EXITS FROM MAIN PROGRAM COMPLETELY.  ********
C*************************************************************************
C
      SUBROUTINE EXIT
      TYPE 220
220   FORMAT(//,10X,' ********************************************',//
     *  ,10X,' **********  ASUPRM IS TERMINATED  **********',//
     *  ,10X,' ********************************************',)
      END
```

REFERENCES

[1] D.L. Moffatt, J.D. Young, E.K. Walton, W. Leeper "Resonant Structure NCTR", ESL Technical Report 714190-2, January 1983.

[2] J.S. Chen, E.K. Walton, "The Ohio State University NCTR Data Base File Structure", ElectroScience Laboratory Technical Report 714190-1, October 1982.

[3] E.K. Walton and J.D. Young, "The Ohio State University Compact Radar Cross Section Measurement Range," submitted to IEEE Trans. on A. and P., May 1984.

END

FILMED

2-86

DTIC